
LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO
SÓLO PROGRAMADORES
ÑO VI. 2ª ÉPOCA NÚMERO 65
UNA PUBLICACIÓN DE REVISTAS PROFESIONALES S.L.
975 Ptas. • 995 Escudos portugueses • 5,86 € (IVA incluido)
LOS MEJORES COMPONENTES A FONDO
00065
8 413042 303299
ACTIVEX
VCL
JAVABEANS
CD
CON MÁS DE
120
ROGRAMAS
BASES DE DATOS
Aplicaciones de Bases de datos con Delphi (II)
LINUX
MP3 (I): Introducción y características
INTERNET
La tecnología ASP (y VI): ASP, XML y XSL
JAVA
Firmado de Applets (I)
PROGRAMACIÓN WEB
Aplicaciones JAVA con acceso a BBDD (III)
ENTORNOS DE DESARROLLO
Visual C++ y MCF (II)

Universe Linux™

La alternativa que su empresa estaba esperando

Linux está en boca de todos: potencia, versatilidad, seguridad con mayúsculas, economía. Y todos se hacen las mismas preguntas. ¿Debo cambiar a **Linux**? ¿Es cierto que **Linux** es más fiable en la empresa que Windows NT, 98 o Windows 2000? ¿Es cierto que **Linux** puede convivir con Windows intercambiando información de forma transparente? ¿Puedo ahorrar con soluciones basadas en **Linux** hasta el 60% del precio en sus programas equivalentes de Microsoft? ¿Por qué más del 65% de los servidores de Internet corren bajo **Linux**? ¿Quién me dará el soporte y la garantía necesarios?

La respuesta a todas las preguntas se llama ***Universe Linux***

Universe Linux es la primera empresa de nuestro país dedicada exclusivamente a ofrecer soluciones basadas en **Linux** para usuarios y empresas: distribuciones **Linux**, utilidades, consultoría, instalaciones a medida, mantenimiento, formación... Todo lo que Vd. necesite y al mejor precio del mercado.

Puede que aún no sepa lo que ***Universe Linux*** puede hacer por Vd. Le invitamos a conocer nuestros productos con una demostración gratuita en nuestra sucursal de Madrid. Llámenos para concertar la demostración. Infórmese sin compromiso llamando al **91-356 69 08** o visitando nuestra página web:

http://www.u-linux.com

Llame hoy mismo. *Por fin* en LINUX hay alguien que RESPONDE.

Linux PYMES

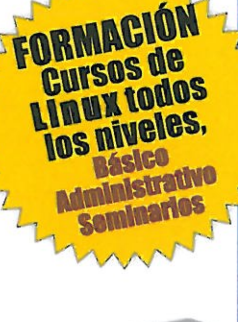
La distribución pensada para empresas
Pvp. recomendado 9.900 pts.

Por fin, la primera distribución creada para las PYMES: robusta, fiable, sencilla de instalar. Incluye: StarOffice, escritorio KDE, manual de instalación paso a paso y, como oferta de lanzamiento, GRATIS el programa LINUX FAX SERVER, valorado en 14.900 pts. (*)

Características:

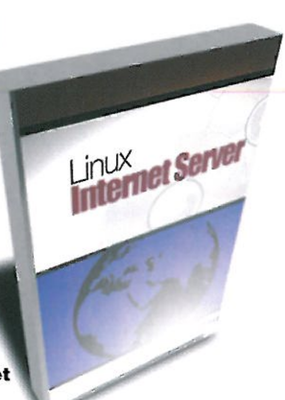
- Instalación en modo gráfico con pantallas de ayuda
- Configuración X completa antes de la instalación de paquetes
- Flexibilidad en los modos de instalación Server y Workstation
- Utilidades de automontaje
- XFree 3.3.5-3, Kernel 2.2.12-20, KDE 1.1.2, GTK, etc.

(*) Oferta válida hasta el 1 de abril del 2000.



Linux INTERNET SERVER

El servidor más fiable
Pvp. recomendado 49.950 pts.



El servidor de Internet pensado para su empresa incluye:
***Servidor de correo electrónico, páginas Web y FTP.**
***Servidor de proxy y caché.**
***Gestión y mantenimiento de listas de correo.**
***Alojamiento del Dominio y Dirección IP fija.**
***Gestión de usuarios y autentificación de los mismos en la entrada al sistema.**
***Control y visualización de usuarios conectados.**
***10 niveles de seguridad en el servidor de páginas Web local.**
***Programación de eventos de conexión con Internet.**
Ver demostración en http:// www.u-linux.com/demo

Linux FAX SERVER

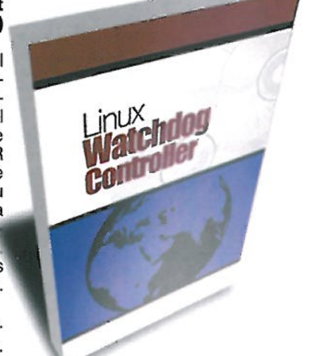
Ahorre enviando todos sus fax por la intranet
Pvp. recomendado 24.900 pts.

Se acabaron las colas para enviar un fax. El primer servidor de fax para Linux selecciona entre los tramos horarios más económicos de su operador de comunicaciones y se adapta a ellos, economizando en cada envío. Evita desplazamientos dentro de la propia empresa ahorrando tiempo y molestias al utilizar la intranet para acceder al servidor. Crea las estadísticas necesarias para analizar los gastos por cada usuario. Economía y comodidad garantizadas en sus comunicaciones.

LINUX WATCHDOG CONTROLLER

El guardián de su empresa en Internet
Pvp. recomendado 69.950 pts. (*)

El primer sistema de CONTROL TOTAL sobre Internet en su empresa. Limitación en el tiempo de conexión, listas de direcciones, chat, en general los puertos más representativos del sistema. Estadísticas de conexión por usuario, por sites accedidas, por distribución del tiempo según operador. Limitaciones individuales o por grupos de usuarios. El sistema corta la conexión cuando detecta que el usuario ha sobrepasado los límites de sus privilegios y le envía un mensaje al supervisor. LINUX WATCHDOG CONTROLLER es una herramienta imprescindible en la empresa moderna para el control del uso de Internet. El 90% de los empleados de las empresas con acceso a Internet desde su puesto de trabajo navegan con fines personales: chats, compras, etc. Esta herramienta erradica el problema realizando auditorías individualizadas de cada usuario, ya que almacena direcciones accedidas, tiempo de estancia, información recibida y enviada, etc. LINUX WATCHDOG CONTROLLER es una utilidad imprescindible para los Administradores del Sistema y Jefes de Personal. Pruébelo sin compromiso.



(*) Hasta 10 puestos de trabajo. Consultar según necesidades superiores.
Los precios no incluyen IVA.

Universe Linux le brinda las mejores opciones para **hacer negocio:**

Linux Training Center

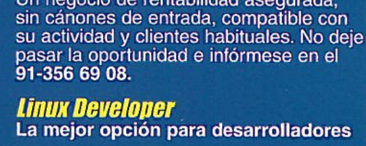
Formación a todos los niveles

Si dispone de una academia de informática Vd. puede tener el LINUX TRAINING CENTER oficial de su población.
Universe Linux formará a su personal y le proporcionará todo el material didáctico necesario y la certificación LINUX TRAINING CENTER. Además, le incluiremos en nuestra Lista de Centros Autorizados en todas las acciones publicitarias promocionando su negocio. (*)
La más amplia gama de cursos para todos los niveles y necesidades en formación: Introducción a Linux, Administración de Sistemas, Instalación y configuración de Servidores web, Ofimática en Linux, etc.
Sistema de franquicias con número limitado de licencias por provincia.

(*) Más de 250.000 impactos cada mes en prensa especializada y general.

Linux Solution Provider

Una apuesta segura para emprendedores. Servicios en Linux

Si es Vd. socio o propietario de una empresa de Servicios Informáticos puede obtener para su provincia la representación en *exclusiva* de los productos **Universe Linux** y la certificación **LINUX SOLUTION PROVIDER** para su empresa de servicios, distribuyendo, instalando y dando soporte a la más amplia gama de productos Linux, con los mejores márgenes comerciales del mercado. Además, contará con el asesoramiento de nuestros técnicos y ayuda on line.

Un negocio de rentabilidad asegurada, sin cánones de entrada, compatible con su actividad y clientes habituales. No deje pasar la oportunidad e infórmese en el **91-356 69 08.**

Linux Developer

La mejor opción para desarrolladores

Universe Linux ofrece a los profesionales independientes del mundo de la programación la posibilidad de adquirir todos los conocimientos para trabajar en el mundo Linux. Incluye el material didáctico y el software necesario a un precio excepcional. Sólo para desarrolladores.

Y también: STAROFFICE, APPLIXWARE, ANTIVIRUS, bases de datos relacionales, servidores de ficheros, y mucho más.

Distribuidores

Universe Linux S.L. España ofrece las mejores condiciones para distribuidores, con descuentos desde el 25% hasta el 40% según tramos de compra. Infórmese sin compromiso en el teléfono: (91) 356 69 08.
Si desea solicitar una demostración gratuita de alguno de los programas, póngase en contacto con nosotros.

Universe Linux S.L. • C/ Conde de Vilches 21 • 28028 MADRID • Teléfono: (91) 356 69 08 • Fax: (91) 355 28 95 • ***http://www.u-linux.com***

SÓLO PROGRAMADORES

Número 65
SÓLO PROGRAMADORES
es una publicación de
REVISTAS PROFESIONALES S.L.
Editor
Agustin G. Buelta
Director
Javier Amado Buiza
Coordinador Técnico
Eduardo De Riquer Frutos
Coordinadoras de Redacción
Gema Romero Moreno-Manzanaro
Cristina Peña del Pozo
Colaboradores
Constantino Sánchez, Juan Luis Ceada,
Javier Sanz, Adolfo Aladro,
Enrique de la Lastra,
Jordi Agost, Vicente A. Sánchez Werner
Maquetación y Tratamiento de Imagen
Paco Risco
Consultas técnicas
atecnica@virtualsw.es

Publicidad
Tel.: (91) 304 78 46
Mariano Sánchez (Barcelona)
Tel.: (93) 322 12 38
Pepín Gallardo (Barcelona)
Tel.: 617 09 36 68

Suscripciones
Rosa Tabares
Tel. (91) 304 87 64 Fax: (91) 327 13 03

Preimpresión
Grebe
Impresión
I. de Impresión
Distribución
Motorpress Ibérica



Depósito legal: M-26827-1994
ISSN: 1134-4792
PRINTED IN SPAIN
COPYRIGHT 31-5-2000
Precio en Canarias, Ceuta y Melilla: 938 ptas. sin I.V.A.
Con sobretasa aérea: 975 ptas.



Asociación Española de Editoriales de Publicaciones Periódicas

EDITORIAL

Componentes al poder

A lo largo de toda la andadura de Sólo Programadores los lenguajes y entornos han crecido extensiblemente tanto en versiones como en nuevas apariciones. Muchos de vosotros hacéis muchas preguntas sobre los componentes como: ¿Cuáles son mejores?, ¿para qué tipo de aplicaciones que realizó? ¿cuál debo utilizar?, ¿cuál me ofrece mayores posibilidades? Y por fin este mes tenéis todas las respuestas a todas vuestras preguntas.

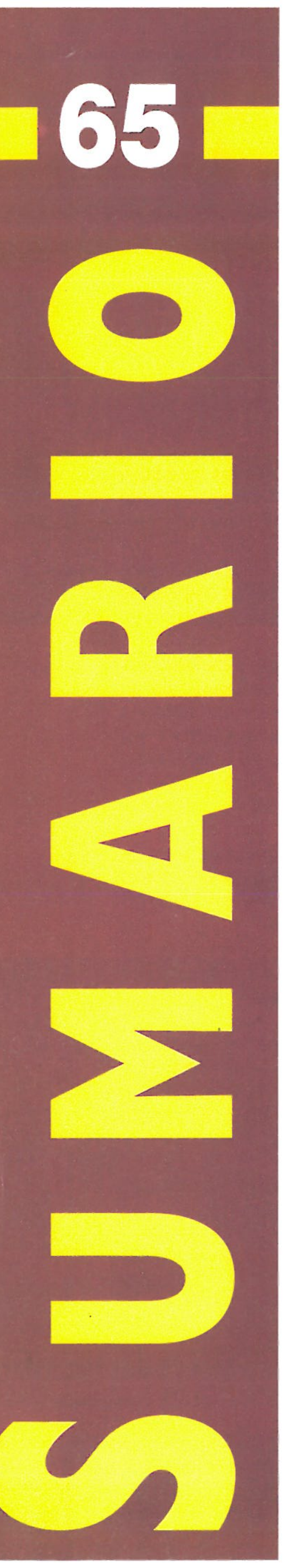
Tres de nuestros mejores colaboradores han aportado al máximo sus conocimientos y han puesto su granito de arena analizando de un modo muy exahustivo los tres mejores componentes del momento (a nuestro entender): *JavaBeans, ActiveX* y *VCL*. Os mostramos los resultados en 17 páginas de la revista. Un análisis que no tiene ningún desperdicio y que os ayudará a conocer las ventajas y desventajas de estas tres competitivas tecnologías.

Aunque desde hace algún tiempo se viene hablando del formato de compresión de audio *MP3,* este mes hemos querido comenzar una serie donde explicaremos desde los conceptos más básicos hasta las últimas tecnologías. Pasearemos por todo el *software* y los lenguajes que nos permiten trabajar con este tipo de formato, pero en un entorno muy específico, el que nos ofrece *Linux*. También comenzamos este número una nueva serie sobre firmado de *Applets*.

Además de todas las continuaciones de los artículos ya empezados como es el caso de: Aplicaciones de Bases de Datos con *Delphi* (II), La Tecnología *ASP* (y VI): *ASP, XML* y *XSL,* Aplicaciones *Web* con acceso a *BBDD* basadas en *JAVA* (III), *Visual C++* y *MFC* (II) y nuestras secciones fijas.

Esperamos que lo disfrutéis. Nosotros volvemos en el número que viene si el código nos lo permite.

Javier Amado
Director



SÓLO PROGRAMADORES

6 Noticias

En los últimos tiempos el mundo de la programación está sufriendo grandes cambios, si no queréis perderos os recomendamos que leáis con atención las novedades de las que os informamos en estas páginas.

9 Contenido del CD-ROM

Como ya es habitual en nuestra revista os regalamos un *CD* en el que podréis encontrar los listados y fuentes de todos los artículos, junto con los mejores programas y las actualizaciones más importantes. Este mes destacamos: *NetObjects Fusion 5.0, VisualAge for Java, Entry Edition 3.0, Nokia WAP Server y BlowFish Machine 1.0*.

38 Linux

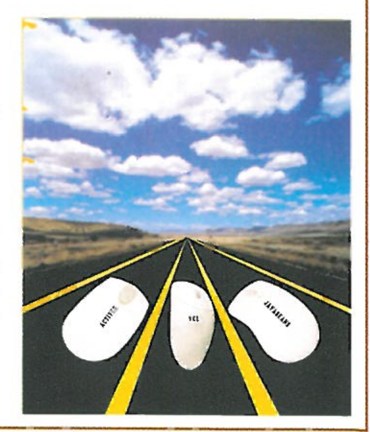
MP3 (I): INTRODUCCIÓN Y CARACTERÍSTICAS



En la actualidad, el formato *MP3* se ha convertido en el medio más popular para almacenar información musical gracias a su reducido espacio, a la alta calidad de reproducción y al pequeño coste de descompresión que requiere. Estas propiedades le han convertido en el rey de *Internet,* desafiando incluso el poder de las poderosas compañías discográficas.

12 Componentes Software

LOS MEJORES COMPONENTES A FONDO



En este número presentamos en un extenso artículo tres de las tecnologías más relevantes en la actualidad, relacionadas con el desarrollo de aplicaciones basadas en componentes. Hablamos, cómo no, de *ActiveX, JavaBeans* y *VCL*. Además de explicar las características y propiedades de cada una de ellas, se analizarán las ventajas y desventajas que poseen con respecto a sus "rivales", desde tres puntos de vista diferentes, de tal forma que el lector encontrará una comparativa que le permitirá elegir la más adecuada en función de sus necesidades concretas.

62 JAVA

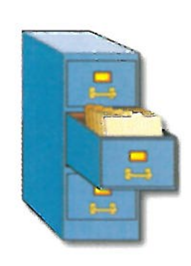
FIRMADO DE APPLETS (I)



¿Quién no ha cargado una página *Web* en la que se ejecuta un *applet Java* y se ha preguntado si ese *applet* podría realizar acciones peligrosas sobre su ordenador?, y en caso contrario, si yo quisiera que las pudiera hacer, ¿qué tendría que cambiar en el *applet?* A estas y a otras preguntas intentaremos responder en la serie que ahora iniciamos.

30 BASES DE DATOS

APLICACIONES DE BASES DE DATOS EN DELPHI 5 (II)



Si en la entrega anterior hicimos algunos comentarios acerca de los alias, en ésta estudiaremos los componentes de acceso y edición de datos. Y ya en el próximo número empezaremos a poner en práctica todos los conocimientos adquiridos en ambas, desarrollando una pequeña aplicación.

46 INTERNET

LA TECNOLOGÍA ASP (y VI): ASP, XML y XSL

El lenguaje *XSL* es el complemento perfecto para las fuentes de datos en formato *XML*, puesto que permite presentar los documentos *XML* de una forma tan sofisticada como se desee. Esto es posible gracias a la capacidad de *XSL* para procesar y tratar los datos *XML*. Hoy en día las páginas *ASP* presentan uno de los entornos más adecuados donde empezar a implantar todas estas tecnologías

80 DUDAS TÉCNICAS

Como siempre en esta sección contestamos a todas vuestras dudas. No os preocupéis si os parecen demasiado complejas o demasiado sencillas, siempre podréis acudir a nuestra dirección de *E-mail:* solop@virutalsw.es. Nosostros intentaremos solucionar vuestros problemas.

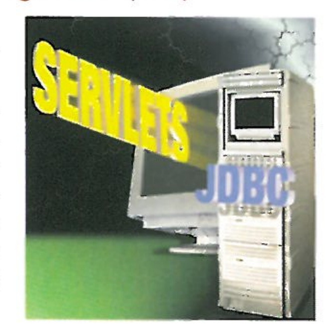
54 ENTORNOS DE DESARROLLO

VISUAL C++ y MFC (II)

Continuamos el desarrollo del programa iniciado en el pasado número para adentrarnos en el control de componentes *ActiveX* dentro de una aplicación. Utilizaremos la caja de texto, un botón de comando y varias fuentes de letra, empezando por un poco de programación.

70 PROGRAMACIÓN WEB

APLICACIONES WEB CON ACCESO A BBDD BASADAS EN JAVA (III)



Es el momento de tomar contacto con la *API JDBC* de *Java,* además de adentrarnos en el proceso de integración con *servlets.* El resultado presentará una aplicación *Web* donde haremos peticiones desde una página *Web* a un *servlet* que comunica a su vez mediante *JDBC* con una base de datos.

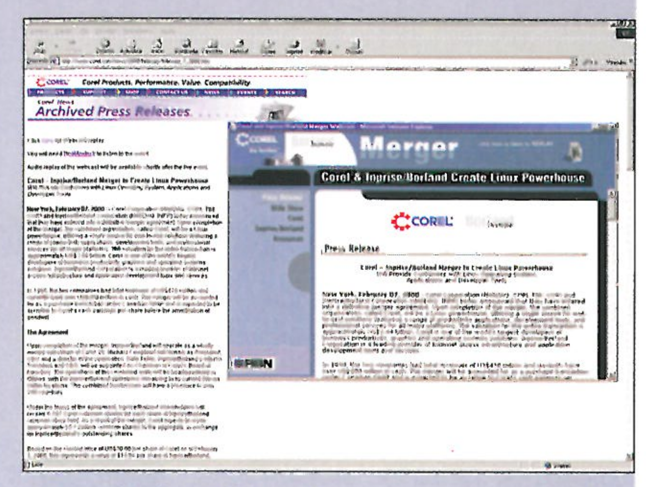
78 LIBROS



Las últimas novedades sobre programación y los libros que nos han parecido más interesantes aparecen en estas páginas, con una sencilla reseña que os ayude a conocerlos mejor y a decidir cuál es el que más os conviene.

COREL E INPRISE SE FUSIONAN

Los gigantes del software *Corel Corporation* e *Inprise,* empresa más conocida por su nombre anterior, *Borland,* se fusionan para aunar sus esfuerzos y reforzar su apoyo a *Linux*. Esta acción se realiza con el ánimo de crear un único grupo empresarial capaz de ofrecer todo tipo de soluciones para cualquier segmento del mercado y bajo las principales plataformas, siendo *Linux* una de las más importantes para la nueva empresa.



Esta nueva empresa, denominada *Corel,* contará con los productos ya creados por las dos empresas matriz, entre los que se encuentran la suite de compiladores *Delphi* y *Builder*, la suite ofimática *Corel Office*, el paquete gráfico *Corel Draw* y la nueva distribución *Corel Linux*.

Esta operación está valorada en 2440 millones de dólares y es una de las más importantes en la historia de la informática, pues supone la unión de dos gigantes.

La nueva empresa pretende convertirse en un centro impulsor de *Linux* por lo que desarrollarán aplicaciones y soluciones para este sistema operativo, además de ofrecer formación y enseñanza para agilizar el paso a esta plataforma. Así mismo la futura empresa seguirá ofreciendo los servicios para *Linux*, *Solaris* y *Windows* que ya ofrecían *Corel Corp*. e *Inprise*, sólo que ahora lo harán de forma conjunta.

Más información en: http://linux.corel.com

ODBC Y JDBC PARA MACINTOSH

Los usuarios de *Macintosh* están de enhorabuena. Poco a poco ven paliada su escasez de herramientas. En este caso se trata de la decisión adoptada por *Merant*, empresa dedicada a las aplicaciones de desarrollo, de trasladar a esta plataforma tanto *ODBC* como *JDBC*, dos aplicaciones punteras en la tecnología de bases de datos.

Ahora, gracias a *Merant*, todos los desarrolladores de *Macintosh* podrán relacionar sus aplicaciones con otras bases de datos como *Filemaker*, *DB2*, Oracle o *Microsoft SQL Server*.

De esta forma se podrá integrar las redes *Macintosh* al comercio electrónico y a los servidores de aplicaciones, combinando productos y servicios integrados de información que incluya a clientes, socios y empleados.

Para más información http://www.merant.com.

NUEVO NAVEGADOR WAP

En la dirección http://www.slobtrot.com/winwap/ la empresa *Slob-Trot Software Oy Ab* ha puesto a disposición de los cibernautas un navegador *WAP* llamado *WINWAP* en su versión 2.2 (1,3 *Mb*). Esta versión emula la visualización de páginas desde un teléfono móvil. Soporta *WML 1.0 and WML 1.1*, permitiendo grabar páginas *WML* como páginas *HTML*. Se trata de un navegador que permite visualizar las páginas *WAP*.

Está claro que este tipo de aplicaciones tienen que proliferar mucho y que aún queda mucho camino por andar, pero estamos en el principio y todavía queda mucho que decir respecto al tema. Por ahora no está de más descargar este navegador y probarlo.



INSTALLSHIELD JAVA EDITION 3.0

Gracias a este producto podremos realizar instalaciones en diversas plataformas con la facilidad de *Windows* y la potencia de *Java*, beneficiando tanto a usuarios como a desarrolladores, a la vez que se reducen los costes de soporte. Además, con *InstallShield* ni siquiera es necesario disponer de la específica *JVM*, si no la localiza el propio programa la instala con la aplicación.

WINDOWS 2000, DISPONIBLE



Windows 2000 es la siguiente generación de *Windows NT* para la empresa. Desde el día 17 de febrero ya están disponibles sus versiones *Professional*, que sustituye a *NT WorkStation* y *Server*, que sustituye a la antigua *NT Server*. *Microsfot* ha anunciado que en breve saldrán el resto de versiones; *Advanced Server* y *DataCenter Server*.

La versión *Professional* propociona, según la propia compañía, la administración de escritorios punto a punto y de redes distribuidas más completa que se puede lograr con otros sistemas operativos. Pues está diseñado para que el cliente y el servidor trabajen juntos, reduciendo los costes de los negocios.

Windows 2000 integra en una sola aplicación los servicios de *Web*, comunicaciones, seguridad ..., además de incluir *Microsoft Internet Explorer* 5.01 en el cliente y los Servicios de Información de *Internet* 5.0 (IIS) en el servidor. Todo ello permitirá a los desarrolladores el construir y distribuir soluciones *Web* y administrar los sitios *Web* o intranets de una forma rápida y fácil. Sin embargo, el integrar todos sus paquetes en el propio sistema operativo la empresa ya está teniendo algunos problemas, empezando por nuevas denuncias por atentar contra la competencia.

Por su parte, el antiguo *NT Server* aparece remodelado en *Windows 2000 Server*, con tres versiones distintas dependiendo de las necesidades de negocio. La opción más básica es lo que ya está en la calle. Como edición separada aparecerá, en breve, *Windows 2000 Advanced Server*, especialmente diseñada para las aplicaciones de negocios y comercio electrónico, pero permitiendo mayor cantidad de usuarios y aplicaciones complejas.

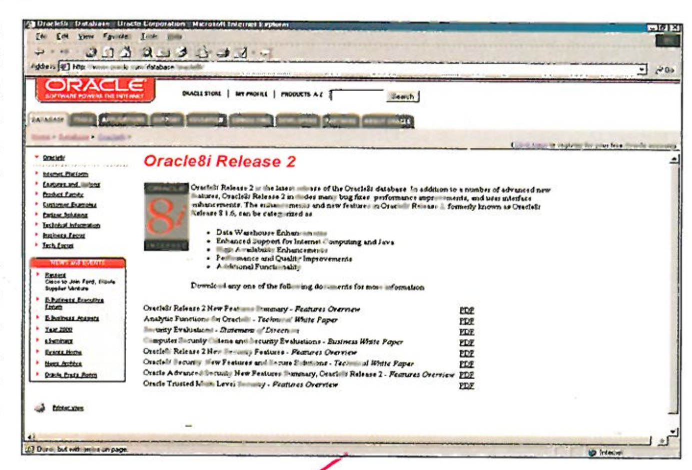
La tercera edición, *Windows DataCenter Server* ampliará las funciones de *Advanced Server* con una mayor capacidad de procesamiento y memoria para cubrir las transacciones de datos en línea, almacenes de datos y proveedores de servicios en *Internet*.

Windows 2000 Professional se vende al precio de 44.500 pesetas y *Windows 2000 Server* para 5 clientes por 154.900 pesetas.

Para más información: http://www.microsoft.com

SEGUNDA VERSIÓN DE ORACLE 8i

Oracle está desarrollando (estará disponible en breve) la segunda versión de *Oracle 8i*. En ésta se pueden encontrar como mejoras destacables: la solución a los *bugs* encontrados en versiones anteriores, mejoras en los sistemas de seguridad, una interfaz de usuarios mejorada, seguridad en multiniveles, etc.



En la página http://www.oracle.com/database/oracle8i/ podréis encontrar toda la información necesaria para aquellos usuarios que la requieran. Para aquellos que necesiten más información en la misma página es posible descargarse ficheros en formato *.PDF* sobre todas las nuevas características y los *White Papers*.

TODO EL SOFTWARE ASP EN UN SOLO PAQUETE



La empresa *Software Artisans*, líder en la creación de aplicaciones para páginas *ASP*, ha reunido en un solo paquete, que se ha denominado *ASPstudio*, todas sus herramientas.

Más de 20 componentes entre los que se encuentran *SA-FileUP, SA-Admin, SA-Archive, SA-FileManagerTX* y *SA-Xfile*. Con ellos se podrá lograr un desarrollo escalable de aplicaciones *Web* listas para insertarse con la plataforma *Windows NT Server*.

Este paquete completo, que está a la venta por 549 dólares, también puede ser adquirido de forma separada.

Para más información:

http://www.softartisans.com

WINDOWS 2000, EL MÁS RÁPIDO

Con *Windows 2000* es posible enviar un *gigabit* por segundo a través de una red *TCP/IP* entre dos estaciones de trabajo a través de un *WAN*, según las pruebas realizadas por diversas compañías tecnológicas en Estados Unidos como *Sony, National Computational Science Alliance* o *Nortel Networks*. Además diversas pruebas realizadas también demostraron altos índices de calidad en un espacio de tiempo de menos de un segundo.

Desde *Microsoft,* con este análisis, pretenden demostrar que su nuevo *Windows 2000* es el servidor más rápido de los que actualmente están en el mercado.

Para más información: **http://www.microsoft.com**

ENTORNOS DE PROGRAMACIÓN

ENTORNOS DE PROGRAMACÓN



EDITORPRO SUITE 1.3

Completo editor de texto que incluye un verificador ortográfico, un diccionario y otro de sinónimos/antónimos. Ofrece funciones avanzadas para nueve lenguajes y múltiples tipos de ficheros.

EDITPLUS 2.00H

Editor de texto para ficheros de *Internet*. Puede sustituir al *Notepad* de *Windows*, pero además ofrece potentes funciones para programadores y diseñadores de páginas *Web* que utilicen *HTML, ASP, Java Script, VBScript, Perl, C, C++* o *Java*.

EUPHORIA PROGRAMMING LANGUAGE 2.2

Lenguaje de programación flexible, rápido y fácil de aprender. Puede crear ficheros ejecutables para *Windows 32 bits* y *DOS 32 bits*. Incluye un depurador de código a pantalla completa, un completo manual de referencia, código fuente y varios programas de ejemplo.

NETOBJECTS FUSION 5.0

Excelente herramienta de creación, publicación y gestión de sitios *Web*. Para generar las páginas incluye *Allaire HomeSite* y un editor visual de la estructura del sitio.



VISUALAGE FOR JAVA, ENTRY EDITION 3.0

Crea aplicaciones compatibles con *Java, applets* y *JavaBeans*. Las aplicaciones creadas se ejecutan en cualquier equipo compatible con *Java* o con cualquier navegador *Java*. Incluye el *Visual Composition Editor* y opciones para compilar clases, incorporar métodos y compilar el trabajo.

WILDTANGENT WEB DRIVER SDK 1.1

Genera elementos *3D* interactivos para navegadores de *Internet*, utilizando un lenguaje estándar basado en *scripts*. Incorpora la potencia de las librerías *DirectX* de *Microsoft* a aplicaciones y páginas *Web*.

LENGUAJES



VISUAL BASIC

BUILD! VB 1.2

Herramienta avanzada de diagnóstico y manejo de errores para *Visual Basic* 5 y 6. Analiza aplicaciones creadas en *Visual Basic* e informa de los errores encontrados utilizando un fichero, un mensaje de correo electrónico y *on-line*. *Build! VB* es totalmente configurable y se integra con el *IDE* de *Visual Basic* y *Sourcesafe*.

FOUNDATION 3.1

Proporciona un manejo de errores activo para *Visual Basic*. Asegura el manejo correcto de los errores de aplicación basados en: tipo de proyecto, componentes, opciones de *firewall*, etc. Incluye un potente asistente *IDE* y un explorador.

VISUAL BASIC SPLASH ANIMATOR 5.0

Genera animaciones para aplicaciones en *Visual Basic*. Permite crear de forma rápida e intuitiva animaciones, seleccionando los elementos y el tipo de animación, y sin tener que escribir código fuente.

JAVA

COCKTAIL 1.1 BETA

Herramienta de generación de código *Java* que representa la estructura de los programas en forma de árbol. Este concepto de estructura de los programas facilita su codificación y mantenimiento.

JAD 1.5.7C

Descompilador *Java* que reconstruye el código fuente original. Es un programa que lee uno o más ficheros de clases *Java* y los convierte en ficheros de código *Java* que puede ser editado y compilado de nuevo.

LINGOGUI 1.1

Colección de componentes totalmente creados en *Java* que añaden a aplicaciones *Web*, paginas *HTML* y servicios *on-line*, además de soporte para 42 idiomas (*National Language Support*).

SOURCE STUDIO 2.0

Entorno de desarrollo integrado para *Java* para crear aplicaciones, *applets*, *Java Servlets*, y componentes *JavaBeans* y *Enterprise JavaBeans* (*EJB*).

HTML

PHP 3.0.14

Lenguaje de scripts *HTML* que facilita la creación de las páginas *Web* dinámicas. Toma el código de *C, Java* y *Perl* y ofrece toda la funcionalidad de un lenguaje de programación.

PLATYPUS JAVASCRIPT EDITOR 1.22

Herramienta de programación capaz de dar a las páginas *HTML* apariencia profesional.

WEBSCRIPTER 3.1.0.0

Editor *HTML* con soporte para *JavaScript*. Incluye menús, botones y asistentes para crear *tags HTML* y un excelente soporte para crear *scripts JavaScript*.

OTROS

BLOWFISH MACHINE 1.0

Implementa el algoritmo *Blowfish* y proporciona opciones de encriptación de alto nivel. *BlowFish Machine* es un servidor *DCOM* de encriptación de ficheros de ámbito internacional que ofrece una seguridad superior a *DES* o Triple *DES*.

CALENDAR OBJX 3.0

Colección de 3 controles *ActiveX* que nos permite visualizar fechas, horas e imágenes en aplicaciones, respectivamente.

HARDINFO LIBRARY 1.5

Colección de clases *MFC* que ofrecen información del *hardware* instalado en un ordenador. Soporta las tecnologías *MMX*, *3Dnow!*, *Frequency*, *Vendor*... y los equipos de los principales fabricantes de *CPU* como *Intel*, *AMD*, *IDT*, *Rise*, *Cyrix* y *UMC*.

INPUT PRO 3.0

Crea pantallas profesionales de introducción de datos utilizando ocho controles. Incluye un control de parámetros *booleanos*, un control memo que muestra grandes cantidades de texto, y seis controles de edición de formato que permiten validar rápidamente los datos independientemente del formato original.

TX TEXT CONTROL 7.0

TX Text es un potente procesador de texto realizado en forma de elementos reutilizables. Proporciona un completo procesador de texto, una aplicación *intranet*, un editor de escritorio.

HERRAMIENTAS Y UTILIDADES

ADVANCED REGISTRY TRACER 1.2

Compara diferentes copias del Registro de *Windows*. Es muy útil para identificar los cambios producidos en un ordenador o comparar la configuración de diferentes equipos.

EASYSOFT CD MENU-GENERATOR 3.00

Crea menús de inicio para *CD-ROM* comerciales. Puede crear una interfaz basada en menús que permita a cualquier usuario instalar o ejecutar los programas de un *CD* de forma sencilla.

ENCRYPT EASY 2.1

Utilidad de encriptación que utiliza dos métodos de entre 67 posibles modos de encriptación. Permite comprimir ficheros.

HEX WORKSHOP 3.02

Editor hexadecimal que incluye un conversor entre formatos hexadecimal, decimal y binario. Realiza las funciones habituales de edición, se integra con el explorador de *Windows* y *Visual* C++, soporta ficheros de varios *gigas* y exporta los documentos a formato *RTF* y *HTML*. Incluye una calculadora hexadecimal que realiza operaciones lógicas y aritméticas.

INFORAPID SEARCH AND REPLACE 3.0C

Utilidad que permite buscar, previsualizar y sustituir cadenas de texto en documentos *HTML* y *RTF*. Utiliza conversores de *Microsoft Word* para buscar ficheros de *Excel*, *Lotus*, *Word*, *WordPerfect* y otros formatos y previsualizarlos con su diseño original.

MICROSOFT WINDOWS MEDIA TOOLS 4.1 BETA

Colección de herramientas de autor y descodificación para producir ficheros de audio y vídeo comprimido bajo demanda. Incluye *Windows Media Encoder, Windows Media Author, Windows Media Plug-In for Adobe Premiere, Windows Media ASF Indexer, Windows Media PowerPoint To ASF*, suplementos para *Windows Media Presenter* y *Publish to ASF PowerPoint*, y varias herramientas de conversión *ASF*.

REDES

ANASIL 2.1

Analizador *LAN* y descodificador de protocolo para redes basadas en *Ethernet*. Entre sus funciones destaca: la monitorización de la utilización global de la red, la distribución del protocolo, el establecimiento de alarmas, los *tests* de transmisión punto a punto (*IP, IPX, Appletalk, NetBEUI*), la captura de paquetes con opciones de filtro y un descodificador de protocolo programable.

DAMEWARE NT UTILITIES 2.8.0.0

Colección de utilidades de gestión remota de servidores *NT* y estaciones de trabajo. Incluye: un visor que muestra los dominios de la red en forma de árbol; un asistente personalizable que facilita los servicios remotos de instalación y desinstalación desde servidores y terminales *NT*; y una utilidad de propiedades de los equipos que muestra información del equipo, la variación de tiempo, el nombre, los *bits* enviados y recibidos, el tiempo de respuesta, etc.

LINKVIEW CLASSIC NETWORK ANALYZER 7.0.0

Analiza redes *Ethernet, Fast Ethernet* y *Token Ring*. Además incluye *Examine*, un descodificador de protocolos que soporta 350 de ellos. Crea informes sobre el tráfico, el protocolo y la captura de fotogramas, y puede avisar de determinados eventos a través de "buscas" o correo electrónico.

NEOTRACE 2.12A

Utilidad que sigue y muestra los nodos de una conexión entre el equipo y una dirección de *Internet*. Detecta posibles problemas de tráfico que dificultan el establecimiento de las conexiones y ofrece los datos en un gráfico o en una tabla. Incluye un comando *Whois*, mejoras en la base de datos de localizaciones, en los gráficos, y un nuevo sistema de ayuda en línea.

NETPROXY 3.05

Servidor *proxy* y *firewall* que permite a los usuarios de una red *LAN* acceder a *Internet* utilizando un equipo *gateway*. Facilita la conexión a *Internet* a través de una única conexión *SLIP*, *PPP* o *TCP/IP*. Utiliza el *RAS* (*Remote Acces Service*) de *Windows*, que permite establecer la conexión con el proveedor de servicios y desconecta automáticamente cuando nadie está utilizando *Internet*, para disminuir el gasto telefónico. Además ofrece protección *firewall*, que impide el acceso a nuestra red a usuarios que no cuenten con autorización para ello.

REMOTEEXPERT 32 V1.50

Programa para controlar y trabajar con otro ordenador en tiempo real a través de módem. Permite transferir ficheros, imprimir en impresoras remotas, mantener sesiones de *chat*, establecer opciones de seguridad, conectarse a una red *LAN*, recibir y realizar llamadas desde el libro de direcciones, etc.

THE NOKIA WAP SERVER 1.0.1

Permite a las compañías proporcionar acceso móvil a información y servicios disponibles en *intranets* privadas y otros sistemas de información. Además de implementar funcionalidad *gateway* entre redes sin cable y servidores de *Web* de compañía, *Nokia WAP Server* trabaja como una plataforma servidor abierta para añadir contenidos, aplicaciones y conectores a otros sistemas de información.

DOCUMENTACIÓN/ TUTORIALES

BEGINNER'S GUIDE TO CGI SCRIPTING WITH PERL

Guía de programación en *Perl* para programadores principiantes. El tutorial ofrece suficientes explicaciones para llegar a ejecutar *scrips CGI* en un servidor de *Web* bajo *UNIX*.

HYPERTEXT GUIDE 2.0

Completo tutorial de programación en *HTML*. Explica de una manera gráfica los principios del lenguaje *HTML*, el modo de dar formato del texto, la visualización de imágenes, los caracteres especiales, la creación de enlaces, elementos multimedia, formularios, tablas, *frames*, mapas de imágenes, fondos estáticos y de marcas de agua, *META tags*, conjuntos de caracteres, instalación de *scripts* y *applets Java* e inclusión de música. El que no sepa programar en *HTML* es porque no quiere.

WAP 1.2 SPECIFICATION

Especificaciones del protocolo *WAP* (*Wireless Aplication Protocol*) en formato *PDF*. *WAP* es un Protocolo para Aplicaciones Sin Hilos que permite el acceso y navegación por *Internet* desde teléfonos móviles o agendas electrónicas de bolsillo (*PDA´s*). Una nueva tecnología se está poniendo muy de moda.

WEB RESOURCES TUTORIALS 6.4

Colección de tutoriales sobre programación de *JavaScripts*, *cookies HTML* y hojas de estilo. Explica cómo crear una docena de efectos especiales, ofrece diversos *scripts* de ejemplo y un completo tutorial para incluir nuestra página *Web* en motores de búsqueda en posiciones ventajosas.

IMPRESCINDIBLES

ANTIVIRUS

AntiViral Toolkit Pro 3.0.129
AntiViral Toolkit Pro 3.0.129
McAfee VirusScan 4.0.3
Panda Antivirus 6.14.00 Platinum

GRÁFICOS

Icon Bank 4.0 Gold Edition Web Edition / Desktop Edition
MicroAngelo 98 v4.77
Paint Shop Pro with Animation Shop 6.02
Reptile 2.0
SureThing CD Labeler 2.0
Ultra Fractal 2.04
Xara WebStyle 1.2

INTERNET

Añadir Pro 4.01.003
Copernic 2000 v4.1
Cuentapasos 3.77
CuteFTP 3.5.6
Dial-Up Magic 1.8
Eudora Light 3.0.6
Free Agent 1.21
GetRight 4.1.2
Go!Zilla 3.5
ICQ 99b beta 3.19 build 2569 Rev.A
i.Share 3.5
MIRC 32 5.7
URL Organizer 2.3.2
WebZIP 3.07
WinGate 3.0.5

MULTIMEDIA

AudioCatalyst 2.01
CDH Media Wizard 4.35
COWON Jet-Audio 4.7
MusicMatch Jukebox 4.5
Sonique 1.30
WinAMP 2.50e

NAVEGADORES

Internet Explorer 5.01
NeoPlanet 5.1
Netscape Communicator 4.7
Opera 3.61

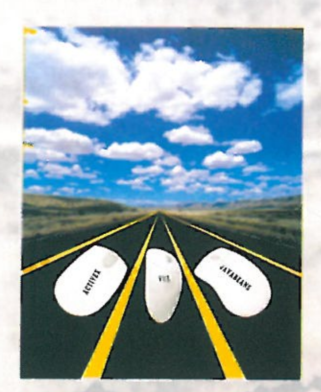
PROGRAMACIÓN

Decafe Pro 3.6
Free Pascal v0.99.14
Hackman 4.02
Help & Manual 2.25
InstallConstruct 3.3
Java Development Kit 1.2.2-001
UltraEdit Professional Text/HEX Editor 7.00b
Windows Registry Guide 1.3
WinHex 9.11

UTILIDADES

Adobe AcrobatReader 4.0
Babylon Translator Spanish 2.2
CallCenter 3.5.8
DirectX 7.0a
Emergency Recovery System 9.04
Nero 4.0.8.3
SiSoft Sandra 99.8.5.30
WebTrends Professional Suite 4.0b
Where Is It? 2.16
Windows Commander 4.03
WinZip 7.0

ATENCIÓN:
En caso de problemas con el CD-ROM envíelo por correo ordinario, a la atención del SERVICIO TÉCNICO DE SÓLO P., incluyendo en el interior del sobre sus datos personales, a la siguiente dirección:
C/ San Sotero, Nº 5, 1ª Planta, 28037 (Madrid)



ACTIVEX, JAVABEANS y VCL

Los mejores componentes a fondo

AxtiveX: Jordi Agost Moré. *Profesor de Programación/Multimedia de la Universidad de Lleida.*
JavaBeans: Javier Sanz Alamillo. *Ingeniero de Software.*
VCL: Juan Luis Ceada Ramos. *Programador en ARCABE Formación y Servicios Informáticos.*

Presentamos tres de las tecnologías más relevantes para el desarrollo de aplicaciones basadas en componentes. Además de explicar sus características y propiedades se analizarán las ventajas y desventajas con sus "rivales", de tal forma que el lector pueda elegir la más adecuada a sus necesidades.

De un tiempo a esta parte, el desarrollo de aplicaciones ha tenido una serie de cambios significativos que han afectado de manera directa nuestra forma de programar. Por una parte, las herramientas visuales han conseguido reducir los tiempos de desarrollo, permitiendo a los programadores centrarse en el problema que se pretende resolver y no en la interfaz que lo envuelve.

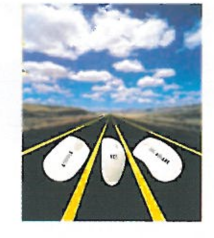
Además, la posibilidad de crear un proyecto mediante el uso de componentes individuales ha permitido reutilizar código ya existente, ya sea por el hecho de haberlo escrito en proyectos anteriores o incluso por terceras partes. En definitiva, el desarrollo de una aplicación se ha facilitado de manera notable, convirtiéndolo en una tarea similar a la conclusión de un puzzle a partir de una serie de piezas individuales, que una vez relacionados de manera apropiada constituyen en último término un conjunto que resuelve los requerimientos planteados al comienzo del desarrollo.

ACTIVEX

UNA PRIMERA APROXIMACIÓN

Un componente *ActiveX* es una unidad de código ejecutable como lo puede ser un archivo *EXE* (ejecutable), o un archivo *DLL* (biblioteca de enlace dinámico), con la única salvedad de que sigue la especificación *ActiveX* para proporcionar objetos. Esta tecnología permite a los programadores ensamblar estos componentes de *software* reutilizables en aplicaciones o servicios.

Así podemos adquirir componentes *ActiveX* que proporcionen servicios genéricos, como por ejemplo en los casos de análisis numérico o elementos de interfaz de usuario, de la misma forma que podemos crear componentes que encapsulen las transacciones de nuestra empresa, y a su vez combinarlos con componentes genéricos. Estos procesos se conocen con el nombre de desarrollo de *software* componente.



DIFERENCIA CON LA PROGRAMACIÓN ORIENTADA A OBJETOS

A veces suele confundirse el desarrollo de componentes *ActiveX* con la programación orientada a objetos. Podemos decir que la programación orientada a objetos es una forma de crear componentes *software* que estén basados en objetos. En cambio *ActiveX* es una tecnología que permite combinar componentes basados en objetos, los cuales han sido creados a partir de muchas herramientas distintas.

Por lo tanto, la programación orientada a objetos se ocupa de crear objetos mientras que la programación de *ActiveX* se ocupa de que los objetos funcionen integrados. Podríamos crear, por ejemplo, con una herramienta orientada a objetos, como es el caso de *Microsoft Visual C++*, objetos para utilizarlos en nuestra aplicación. Otros programadores podrían ampliar si lo deseasen dichos objetos, pero si éstos son encapsulados en un componente siguiendo la tecnología *ActiveX* entonces los podremos utilizar y ampliar con cualquier herramienta de programación compatible con la tecnología *ActiveX.*

Otra confusión habitual es confundir los componentes *ActiveX* con los servidores *OLE*. Si bien es cierto que la nueva tecnologia *COM* (es la tecnología que permite crear componentes *ActiveX* siguiendo el Modelo de objetos de componentes o modelo *COM (Component Object Model))* proviene de la tecnología de servidores *OLE,* existen grandes diferencias entre los dos.

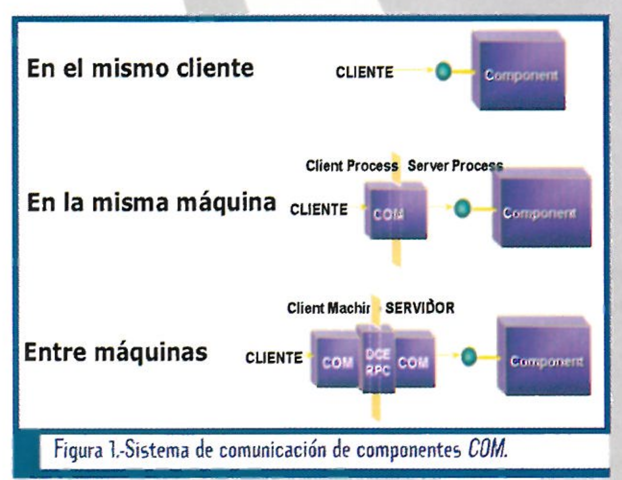


Figura 1.-Sistema de comunicación de componentes *COM*.

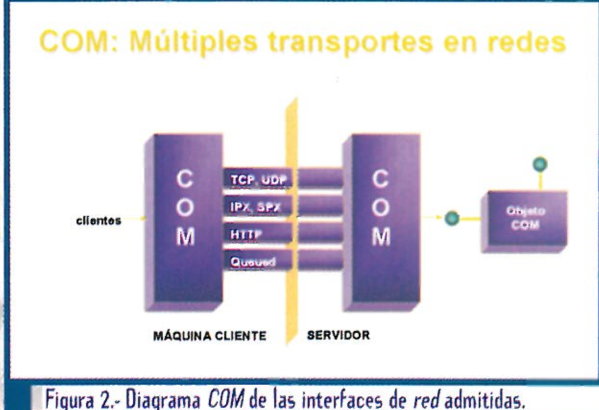


Figura 2.- Diagrama *COM* de las interfaces de *red* admitidas.

DCOM O COM DISTRIBUIDO

Una vez experimentado con éxito todo el sistema de componentes, se llegó a una evolución lógica de tal sistema, creándose así la tecnología *DCOM.* La tecnología *DCOM* es en realidad un protocolo de los objetos *COM* que les permite comunicarse directamente con otros a través de una red, sea ésta de tipo local *(LAN, Intranet, WAN)* o incluso *Internet. DCOM* también es un lenguaje neutral, lo que significa que cualquier lenguaje capaz de realizar componentes *ActiveX* también puede producir aplicaciones *DCOM.*

CARACTERÍSTICAS

A continuación vamos a ir enumerando las principales características de la tecnología COM/DCOM. Resumamos brevemente las principales:

- *Software* más fácil de escribir y reutilizar.
- Amplio abanico de herramientas, lenguajes y servicios.
- Controles componentes y herramientas de servidor.
- Cualquier lenguaje de programación, e integrados dentro de las herramientas de programación.
- Seguridad integrada.
- Ubicuidad.
- Múltiples transportes en Red.

Una de las primeras ventajas con que nos encontramos es que el uso de la tecnología *ActiveX* se ha extendido a prácticamente cualquier lenguaje de programación, con lo que si creamos un componente que funciona en esta tecnología lo podremos usar en lenguajes tan dispares como *Visual Basic, Visual C++, COBOL, FORTRAN, REXX, Delphi, Lingo,* etc.

Derivado directamente de la característica anterior nos encontramos que el *software* escrito a través de esta tecnología es mucho más fácil de reutilizar, ya que no depende íntegramente del lenguaje en que ha sido escrito. Esto ha logrado que muchas empresas se dediquen a desarrollar un amplio abanico de herramientas, servicios y aplicaciones ya que el público objetivo no está limitado por el lenguaje y es muy amplio.

En el transporte en red la información transmitida desde el servi-

dor al cliente (o viceversa) puede realizarse mediante múltiples protocolos como por ejemplo *TCP/IP, IP, IPX, SPX, HTTP..*

También en el tema de la seguridad admite múltiples sistemas de seguridad, entre ellos podemos citar por ejemplo al subsistema de seguridad propio de *NT4*, certificados, *SSL, Kerberos,* e *IPSEC.*

ECHANDO UN VISTAZO HACIA EL FUTURO: COM+

¿Qué es *COM + ó COM plus?* La respuesta es que es la lógica evolución de la arquitectura de componentes *COM. COM+* ha ampliado de manera significativa los servicios, veamos un pequeño resumen de lo que nos deparará esta nueva tecnología:

- *Componentes de cola:* Dichos componentes permitirán a las aplicaciones clientes invocar métodos en componentes COM utilizando un modelo asíncrono. Este proceso es particularmente útil en redes o equipos que no se encuentren conectados.
- *Servicio de publicación y suscripción:* Dicha novedad es un mecanismo que permite que múltiples clientes puedan suscribirse a varios eventos publicados. Cuando el programa que tiene que publicar el evento, lance un evento determinado, el sistema de eventos de *COM+* examinará su base de datos y lo notificará a todos los clientes.
- *Carga Dinámica equilibrada:* Automáticamente divide las peticiones de un cliente entre múltiples componentes COM equivalentes.
- *Plena integración de Microsoft Transaction Server (MTS) dentro de la tecnología COM:* la nueva tecnología soportará el protocolo de transacciones de *Internet (TIP)* y se han ampliado de manera significativa servicios tales como las transacciones, la seguridad y la administración.

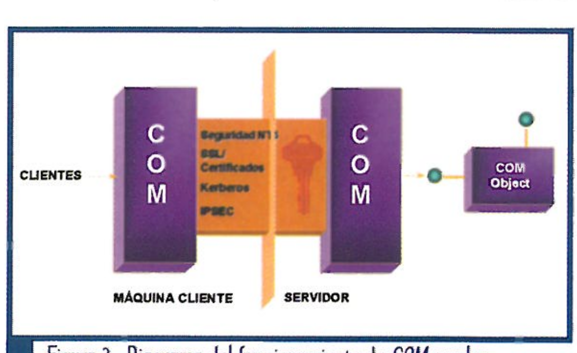


Figura 3.- Diagrama del funcionamiento de *COM* con los protocolos de seguridad.

JAVABEANS

¿QUÉ ES UN JAVABEAN?

De las múltiples definiciones existentes, en la especificación sobre *JavaBeans* de *Sun* encontramos la siguiente:

"Un *JavaBean* es un componente *software* reutilizable que puede ser gestionado mediante herramientas visuales."

Todo esto es cierto gracias a que existe una especificación del componente *software* para *Java*, definida en el *JavaBean API*. Esta especificación sobre *JavaBeans* determina una serie de características comunes a todos los beans que son:

- *Introspection (inspección),* por la que una herramienta visual puede analizar la funcionalidad de un JavaBean.
- *Properties (propiedades),* por lo que se puede definir el estado de un bean.

Los JavaBeans por desarrollarse en Java son multiplataforma

- *Events(eventos),* que define la forma en que los JavaBeans interactúan con los elementos que los utilizan.
- *Persistence (persistencia),* gracias al cual un bean puede gestionar su estado para recuperarlo posteriormente.
- *Deploying (empaquetamiento),* define cómo se realiza el proceso de distribución de componentes.
- *Customization (personalización),* por lo que el usuario puede personalizar la apariencia y comportamiento de un *bean* en una herramienta visual.

Un *bean* no tiene que derivar ni implementar ninguna interfaz especial para su creación como tal, pero sí que debe seguir ciertas reglas y las principales son:

- Constar de un constructor sin argumentos.
- Soportar persistencia, mediante la implementación de la interfaz *Serializable* o *Externizable.*

TIEMPO DE DISEÑO Y EJECUCIÓN

Los *JavaBeans* tienen que poder ejecutarse en dos entornos diferentes. En el primero de ellos, un *bean* tiene que poder ejecutarse en una herramienta visual, es decir, dentro del entorno de diseño. El *bean* tiene que ofrecer a las herramientas visuales informa-



ción sobre su construcción, diseño, etc., de tal forma que el usuario pueda personalizar su *bean* visualmente. Para poder realizar esto, es necesario introducir un conjunto de información en el propio *bean,* como son propiedades de edición y personalización, iconos, etc.

En la Figura 4 se muestra un sencillo *JavaBean* visual.

Pero también debe poder utilizarse en tiempo de ejecución, durante el que no se requiera tanta información y que por tanto, no se necesiten tantos añadidos. Esto nos lleva a que pueden definirse *Javabeans* visuales y no visuales.

JAVABEANS NO VISUALES

Aunque la mayoría de los *beans* tienen presentación gráfica, pueden crearse componentes que no la requieran. Estos tipos de *beans* se denominan no visuales.

Su principal diferencia con los *JavaBeans* gráficos, que son los que normalmente se emplean, es que no incluyen el conjunto de información necesaria para ser utilizados en entornos gráficos, para interactuar con otros componentes, aunque sí deberían poder ser utilizados en entornos gráficos.

Value for label:

Value for textfield:

Apply changes

Figura 4.- Componente visual.

CARACTERÍSTICAS DE LOS JAVABEANS

Ampliemos las definiciones sobre las características de los *beans.*

INSPECCIÓN

Se denomina *introspection* al proceso por el cual una herramienta visual analiza las propiedades, eventos y métodos de que dispone un *bean*. Se realiza mediante el uso de la *API Reflection.* Para ello, se deben seguir una serie de convenciones a la hora de construir el *bean,* de tal forma que se definen las propiedades simples, métodos, eventos, etc.

PROPIEDADES

Las *properties* o propiedades son los atributos que definen la apariencia o comportamiento de un *bean.* Por ejemplo, en un componente tipo botón, podría ser su estado *enable/disable*. Las propiedades pueden ser de lectura/escritura, de sólo lectura o de sólo escritura, en función de los métodos que se definan, esto es, los correspondientes *set/get* que surgen en el proceso de *Introspection.*

Puesto que no es obligaroria una parte visual, se pueden crear beans con diferentes usos

Existen diferentes tipos de propiedades, que vamos a describir.

- *Simple Properties:* Las propiedades simples representan un valor simple y su declaración se basa en definir métodos *set<Propiedad>* y *get<Propiedad>* para todos los tipos excepto el *boolean* que cambia *get<Propiedad>* por *is<Propiedad>*, donde *<Propiedad>* es el tipo definido.
- *Indexed Properties:* Este tipo

TABLA 1. Convenciones de nombrado para las propiedades JavaBeans.

Methods	Public methods
Simple property	public void setPropertyName(PropertyType value); public PropertyType getPropertyName();
Boolean property	public void setPropertyName(boolean value); public boolean isPropertyName();
Indexed property	public void setPropertyName(int index, PropertyType value);public PropertyType getPropertyName(int index); public void setPropertyName(PropertyType[] value); public PropertyType[] getPropertyName();
Multicast events	Public void addEventListenerType(EventListenerType l); public void removeEventListenerType(EventListenerType l);
Unicast events	Public void addEventListenerType(EventListenerType l) throws TooManyListenersException; public void removeEventListenerType(EventListenerType l);

de propiedades representan tablas de valores. Los métodos *get/set* necesitan de un entero como indicador de posición en la tabla.

- *Bound Properties:* Una *bound properties* representa una propiedad que notificará a otros objetos cuándo un valor determinado cambia. Cuando esto ocurre, se genera un evento del tipo *PropertyChangeEvent.* Este evento contiene información sobre el nombre de la propiedad, el valor anterior y el valor actual.
- *Constrained Properies:* Este tipo de propiedad es similar a la anterior, con la diferencia de que cuando un objeto es avisado de que una propiedad ha cambiado, éste puede determinar si el cambio se lleva adelante o se anula (mediante el lanzamiento de una excepción). El proceso se basa en el tratamiento de un evento del tipo *VetoableChangeEvent.*

En la Tabla 1 se muestra un resumen de las convenciones de nombrado para las propiedades.

EVENTOS

Los *JavaBeans* utilizan el modelo de eventos definido en *Java 1.1*. La gestión de eventos se basa en el concepto de objeto-evento *"listener"*. Un objeto que desee recibir un evento construye un *event listener.* El objeto que genera un tipo de evento se denomina *event source*. En el caso de los *beans,* éstos son *event source* y las aplicaciones que los utilizan son *event listener* o simplemente *listeners.*

El *objeto event source* mantiene una lista de los *listener* y consta de unos métodos para añadir y eliminar los *listeners* relacionados. Cuando ocurre un evento, el *event source* notifica a todos los *listeners* registrados esa circunstancia. Esta notificación se realiza invocando un método y pasando un determinado objeto, por lo que los objetos que hayan registrado un evento mediante su *listeners* serán notificados.

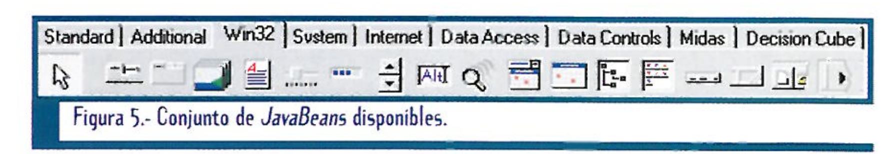


Figura 5.- Conjunto de *JavaBeans* disponibles.

Gracias a la especificación sobre JavaBeans, todas sus características están perfectamente definidas

Los eventos se pueden definir en una herramienta *RAD,* tal y como se aprecia en la Figura 7.

PERSISTENCIA

La persistencia es una operación que permite almacenar un objeto al que se le haya aplicado un proceso de serialización, por lo que se guarda su estado actual y se puede restaurar posteriormente, pudiéndose así utilizar el objeto como si no le hubiera ocurrido nada, por tanto, permanece en el mismo estado en el que se aplicó la persistencia.

La utilización de la persistencia por los entornos visuales se basa en almacenar las propiedades de un *bean* para más tarde actualizar su estado y obtener sus propiedades intactas.

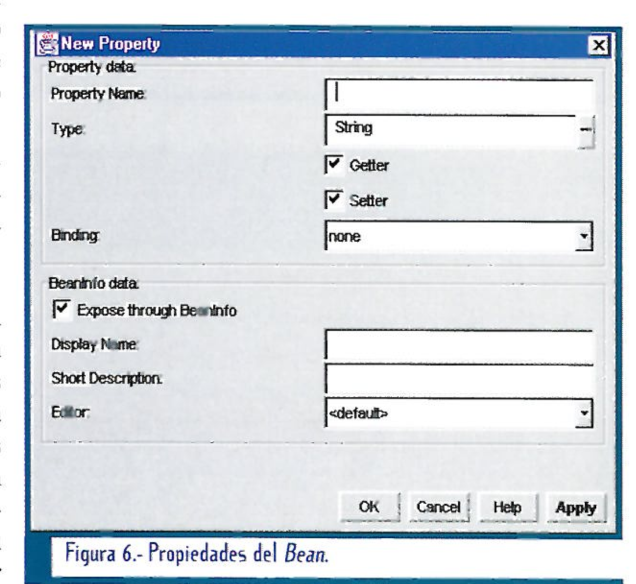


Figura 6.- Propiedades del *Bean.*

EMPAQUETAMIENTO

La especificación sobre *JavaBeans* describe que un *bean* se construirá a partir de un fichero tipo *JAR* con una estructura de fichero tipo *ZIP.* Un fichero *JAR* puede incluir objetos serializados, documentos, imágenes, etc. Pues bien, el proceso de crear un fichero *JAR* se denomina empaquetamiento. El proceso de reconocimiento de un *JavaBean* en un *JAR* se basa en el reconocimiento del fichero *MANIFIEST* que se encuentra en el mismo. Este fichero consta de una línea de información:

```
Java-bean: true
```

en la que como se observa, la entrada *Java-bean: true* identifica el contenido del *JAR*.

INTEROPERACIÓN

Como no es previsible que los desarrolladores abandonen los sistemas que conocen y se pongan a utilizar *JavaBeans, Sun* incluyó

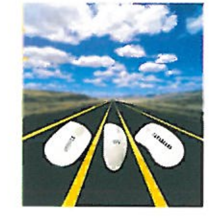


Figura 7.- Gestión gráfica de eventos.

en el *JDK 1.1* para los programadores *Windows* una interfaz entre los *Beans* y los *ActiveX,* que permite realizar desarrollos utilizando estos dos diferentes tipos de componentes, lo que se define como interoperación.

CUSTOMIZATION

La apariencia y comportamiento de un *bean* en una herramienta visual puede ser modificados mediante el uso de editores de propiedades y personalización e información de clases *BeanInfo.*

El *editor de Propiedades* permite al usuario cambiar "el valor" de las diferentes propiedades disponibles. Por ejemplo, el *Layout* del componente.

Los beans son fácilmente utilizables en entornos RAD gracias a sus posibilidades de ajuste (Customization)

Un editor *Customizer,* permite tener un control sobre cómo configurar o editar un *bean,* pudiendo modificar las distintas propiedades. Se puede decir que es una aplicación que se encarga de gestionar el contenido de los *beans.*

Un *bean* puede implícitamente exponer sus propiedades como se ha indicado hasta ahora o realizar una exposición explícita mediante el uso de clases *BeanInfo,* que son utilizadas tanto por el desarrollador como por las herramientas visuales para gestionar el *bean.*

VCL

INTRODUCCIÓN

Hasta la aparición de los lenguajes *RAD (Rapid Development Application),* la programación bajo *Windows* era una tarea tediosa. Había que conocer a fondo la *API* de este sistema y perder mucho tiempo en tareas secundarias (como por ejemplo, en desarrollar el código necesario para situar controles en pantalla). Pero aparecieron lenguajes como *Visual Basic* y *Delphi,* que iban a cambiar por completo el panorama.

Con el lanzamiento de *Delphi 1.0* (a mediados del 95), *Borland* facilitó a los desarrolladores un *framework* conocido como *VCL (Visual Component Library,* Biblioteca de Componentes Visuales). Ésta se compone de un conjunto de clases que permiten desarrollar aplicaciones *Windows* en un tiempo record. Dentro de *Delphi* no se trabaja directamente con las clases, sino que se usan componentes, que vienen a ser la representación visual de una clase.

VCL ha sido desarrollada con el lenguaje Object Pascal

El lenguaje de programación elegido para la creación de la *VCL* fue una evolución de *Pascal, Object Pascal,* que como su nombre indica, es un lenguaje orientado a objetos. Después, con *Delphi 2.0,* la *VCL* se hizo de 32 *bits,* y ganó en velocidad y en robustez.

Figura 8.- JavaBean Info.

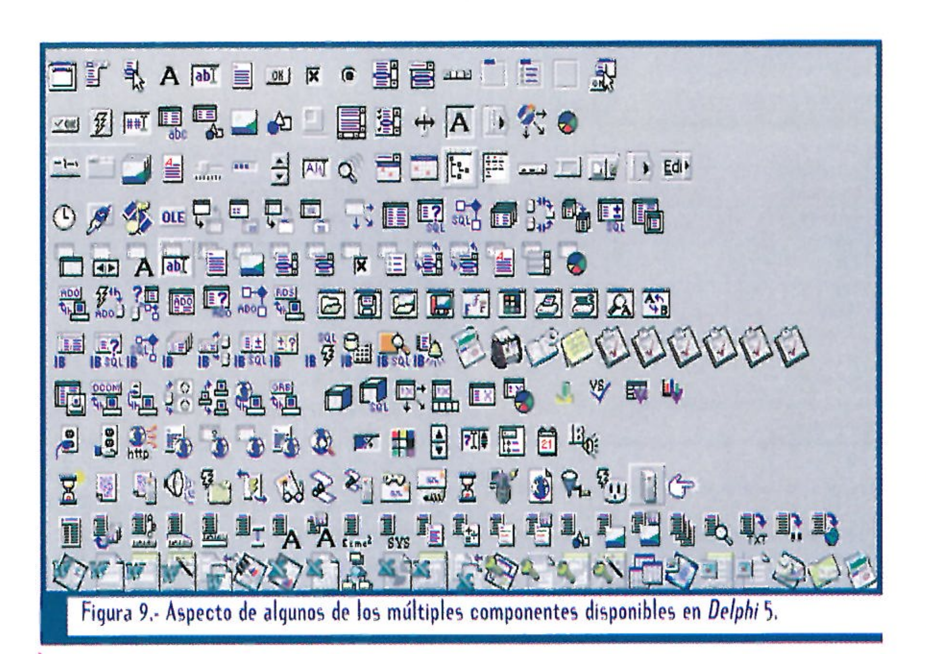


Figura 9.- Aspecto de algunos de los múltiples componentes disponibles en *Delphi* 5.

CARACTERÍSTICAS DE LA VCL

La *VCL* se ha desarrollado en *Object Pascal,* y por tanto, es completamente orientada a objeto. Es decir, soporta la encapsulación, la herencia y el polimorfismo. Es posible desarrollar nuevas clases, bien partiendo de cero, o mediante herencia. Esta es una de sus mayores ventajas. ¿Necesitamos modificar el comportamiento de un componente? Basta con crear una nuevo componente, que herede del anterior, y modificar lo que sea necesario.

Al desarrollador de componentes le interesará saber que desde un mismo entorno de desarrollo (en este caso, *Delphi)* es posible usar los componentes de *VCL,* o bien crear otros nuevos. En ningún momento tenemos que recurrir a otros lenguajes para crear/modificar componentes. También es posible usar *C++ Builder* para desarrollar los componentes (todo un acierto por parte de *Inprise* el hacer que *Delphi/C++ Builder* sean compatibles en este aspecto).

Un desarrollador de aplicaciones debería saber que la *VCL* está diseñada de forma que el programador pueda manipular las clases dentro del *IDE* de *Delphi,* en tiempo de diseño, mientras se está creando la aplicación. Gracias a la retroalimentación, cualquier cambio en cualquiera de las propiedades de un componente, se ve reflejado en su aspecto visual. De esta forma, es posible saber cuál va a ser el aspecto final de un componente en tiempo de diseño.

Incluso en controles de bases de datos, la retroalimentación es inmediata. Si enlazamos un control de rejilla de datos (un *TDBGrid)* con una tabla, automáticamente se crean tantas columnas en el ala rejilla *(grid)* como campos tenga la tabla. Además, en el caso de que se encuentre activa, se mostrarán los datos. Si cambiamos la máscara de visualización de uno de los campos, automáticamente cambia el formato de visualización del campo en la rejilla.

VCL está orientada a objetos (permite encapsulación, herencia y polimorfismo)

Otra de sus características es la velocidad de ejecución. Los componentes *VCL* se integran directamente dentro de los ejecutables producidos por *Delphi* y por *C++ Builder.* Las llamadas a métodos, propiedades y eventos se realizan de forma muy eficiente. Además, el compilador de *Delphi* es uno de los más avanzados del mercado, por lo que el código generado está muy optimizado. En un único ejecutable tenemos todo lo necesario para que nuestra aplicación funcione (nada de *DLL's* u otros ficheros adicionales).

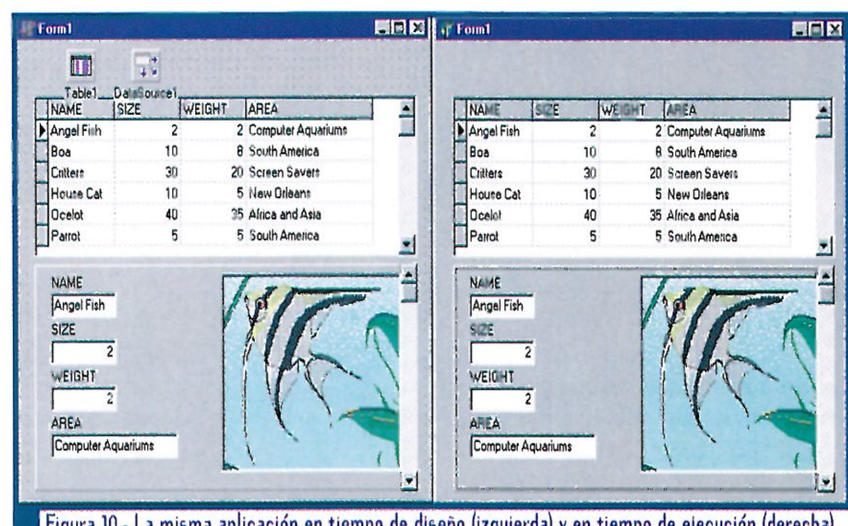
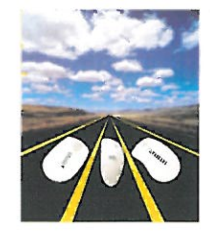


Figura 10.- La misma aplicación en tiempo de diseño (izquierda) y en tiempo de ejecución (derecha).



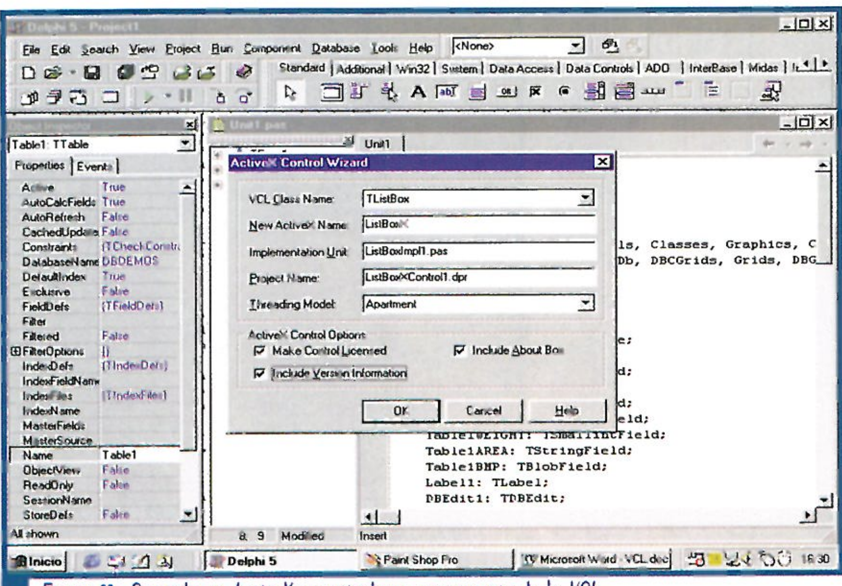


Figura 11.- Creando un *ActiveX* a partir de un componente de la *VCL*.

Esto tiene también sus inconvenientes, ya que el tamaño de los ejecutables es mayor que en otros lenguajes. Un simple ejecutable, que muestre una ventana con un botón puede ocupar fácilmente 400 *Kb*. Esto es debido a que se incluyen las clases básicas de la *VCL*. A partir de ahí, añadir nuevos componentes a la aplicación hace que el tamaño del ejecutable crezca de una forma más mesurada.

Para solucionar este problema, podemos recurrir a la compilación con paquetes. Es posible organizar los componentes de terceros que usemos (o desarrollemos) en un único paquete. Después, activamos el enlace *(link)* usando paquetes y conseguimos dos cosas: el tamaño del ejecutable disminuye hasta 10 veces y en caso de que haya varias aplicaciones que usen el mismo paquete, todas comparten la misma zona de memoria donde el paquete se encuentra ubicado. Es decir, se ahorra memoria y las aplicaciones cargan más rápido. El inconveniente es que nuestros ejecutables deben ir acompañados de ficheros auxiliares.



Otra característica adicional es que es posible, mediante un sencillo asistente, transformar cualquier componente de *VCL* en un control *ActiveX*. Así, podríamos desarrollar un componente *VCL*, que usaríamos en *Delphi* y *C++ Builder*, y si necesitamos usarlo en *Visual Basic* (o cualquier otro lenguaje que soporte *ActiveX)*, en cinco minutos tendríamos listo nuestro control *ActiveX*. Por supuesto, en *Delphi* se pueden usar controles *ActiveX*, con lo que podríamos hacer pruebas con nuestro nuevo control.

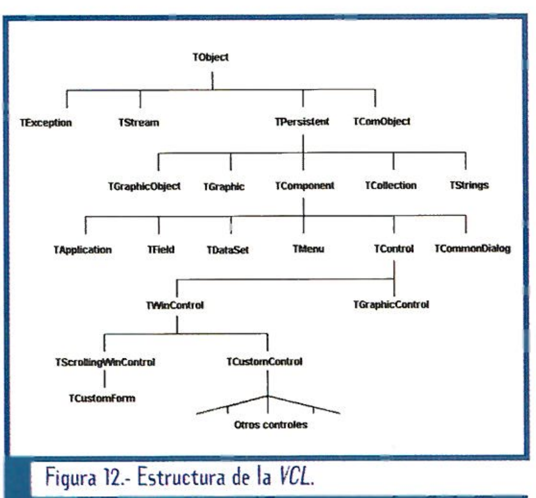


Figura 12.- Estructura de la *VCL*.

La velocidad de los componentes de la VCL es envidiable

A partir de *Delphi 5*, es posible crear un nuevo componente a partir de varios, agrupándolos en un *frame*, que desde ese momento será tratado como un sólo componente (permitiendo, por ejemplo, crear componentes heredados de él). Un ejemplo sería el típico control de edición con un botón asociado. Podríamos crear, de forma visual, un nuevo componente llamado *TEditConBoton*, y a partir de él, otro llamado *TEditConBotonyLabel*, y así sucesivamente.

La compilación con paquetes permite ahorrar espacio en disco y memoria

Todas estas características han hecho que mucha gente se haya lanzado a crear sus propios componentes, por lo que actualmente existen componentes para casi todo lo que podemos imaginar (un ejemplo: HYPERLINK http://ftp.uniovi.es/pub/delphi http://ftp.uniovi.es/pub/delphi). Por supuesto, la combinación *Delphi/C++ Builder* incluye una extensa colección, por lo que en la mayoría de los casos, no será necesario recurrir a componentes desarrollados por otras personas.

TABLA 2 Principales ventajas e inconvenientes de VCL

Ventajas	Inconvenientes
Código rápido y optimizado	No es multiplataforma (aunque esto puede cambiar gracias a Kylyx, el Delphi para Linux)
Totalmente orientado a objetos. Permite encapsulación, herencia y polimorfismo.	El tamaño de los ejecutables es mayor (aunque puede solucionarse con la compilación con paquetes)
Es posible crear controles ActiveX a partir de componentes VCL	
Retroalimentación inmediata: lo que vemos en tiempo de diseño se corresponde con el resultado final	
Es posible ampliar/modificar la VCL desde el propio Delphi/C++ Builder	
Los componentes van dentro del propio ejecutable, sin necesitar ficheros externos	
Es posible ahorrar espacio y memoria agrupando los componentes en paquetes. Compatible Delphi/C++ Builder	

Por último, es posible adherir el código fuente completo de *VCL,* por lo que podemos modificar a nuestro antojo su comportamiento, aunque no es aconsejable.

ESTRUCTURA DE LA VCL

Veremos ahora unas nociones básicas sobre la estructura de la *VCL.* Como hemos comentado, la *VCL* no es más que una jerarquía de clases. Los componentes, que es con lo que el desarrollador de aplicaciones trabaja, son parte de ella. Vienen a ser como los ladrillos para el constructor. Añadiendo ladrillos con un determinado orden, poco a poco iremos construyendo una casa.

Todas y cada una de las clases de *VCL* tienen como base a la clase *TObject* (ver Figura 11), que no es más que una clase abstracta que encapsula aspectos como la creación, destrucción de las clases y el manejo de mensajes.

Los componentes de *VCL* descienden de la clase *TComponent.* Los componentes no son más que objetos que el usuario inserta en los formularios, y que puede modificar en tiempo de diseño. Los hay muy básicos, como el componente *TLabel* (para visualizar etiquetas), y muy complejos (por ejemplo, *TDBChart,* dedicado a la visualización de gráficos a partir de un conjunto de datos).

COMPONENTES VISUALES

Son aquellos que se muestran en tiempo de diseño y en tiempo de ejecución. Podemos dividir este grupo en dos: *"windowed controls"* y *"non-windowed controls".* Los primeros están asociados a una ventana, y por tanto, poseen su propio *handle.* Por lo tanto, pueden recibir el foco, consumen recursos del sistema, y pueden

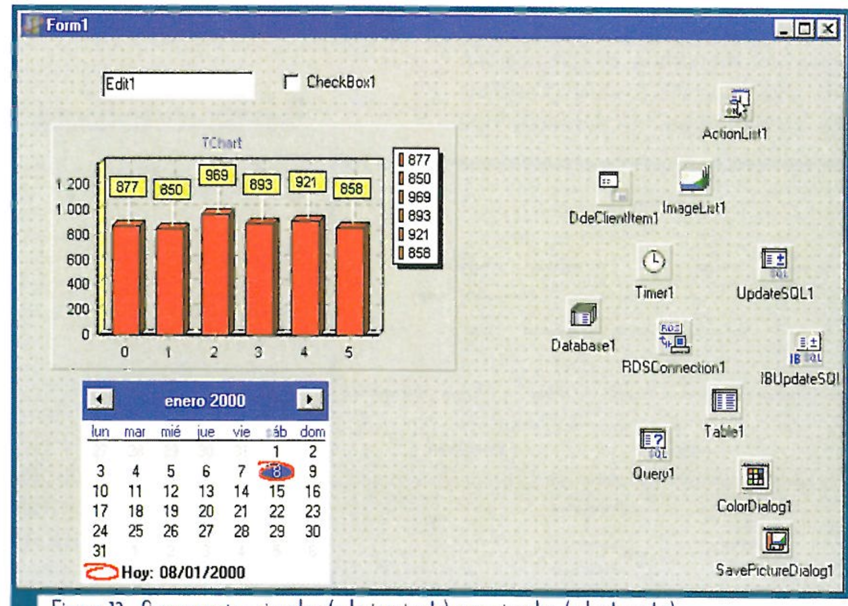


Figura 13.- Componentes visuales (a la izquierda) y no visuales (a la derecha).

ser padres de otros controles (por ejemplo, un *TPanel*, que es un control contenedor, esto es, puede contener en su interior a otros controles).

Los segundos son controles visuales que no reciben el foco, y por lo tanto, el usuario no puede interactuar con ellos. Son útiles para mostrar información al usuario sin consumir recursos del sistema. En este grupo tenemos al componente *TLabel* (para mostrar etiquetas) y a *TShape* (para dibujar figuras geométricas).

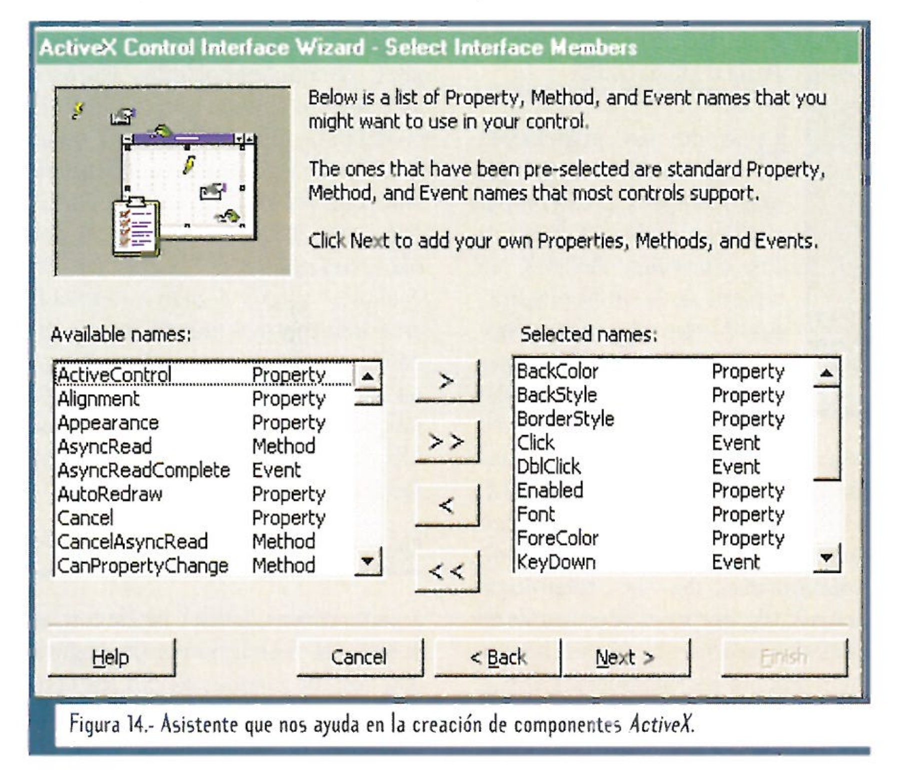


Figura 14.- Asistente que nos ayuda en la creación de componentes *ActiveX*.

COMPONENTES NO VISUALES

Son componentes que sí aparecen en tiempo de diseño (normalmente representados por un icono identificativo), pero no lo hacen en tiempo de ejecución. Un ejemplo de este tipo de componentes es *TOpenDialog*, que sirve para mostrar el cuadro de diálogo de apertura de ficheros.

Este tipo de componentes son muy utilizados para encapsular funciones de la *API de Windows*. Por ejemplo, podemos crear un componente que encapsule todo la *API* de comunicaciones por el puerto serie. En tiempo de diseño, introduciríamos uno o varios de estos componentes, a los que podríamos cambiarles sus propiedades principales (como por ejemplo, puerto *COM*, velocidad, etc.).

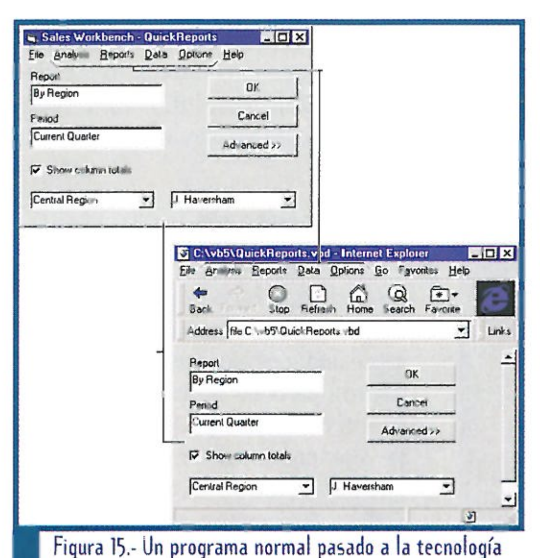


Figura 15.- Un programa normal pasado a la tecnología *ActiveX* para así visualizarlo en un navegador.

COMPARATIVA 1

ACTIVEX Vs. JAVABEANS y VCL

GENERAL

La tecnología *ActiveX* es un estándar dentro de la plataforma *Windows*. A continuación veremos una comparativa con otras de las plataformas presentes en el panorama actual.

ACTIVEX Vs. JAVABEANS

JavaSoft por un lado y *Microsoft* por otro han hecho que las tecnologías *JavaBeans* y *ActiveX* se encuentren bastante enfrentadas entre ellas, aunque por el momento esto no haya significado poner en peligro ninguna de estas tecnologías. A continuación se muestra una lista con los principales apartados que se van a analizar.

- Multiplataforma.
- Dependencia del lenguaje de programación.
- Seguridad.
- Compilación.
- Código existente.

MULTIPLATAFORMA

Una de las principales ventajas de la tecnología que acompaña a los componentes *JavaBeans* frente a los controles *ActiveX* se refiere a la multiplataforma. Mientras que la tecnología *ActiveX* solamente puede ejecutarse bajo plataformas *Windows*, la primera puede hacerlo en plataformas tan dispares como *Windows*, *UNIX*. Esto no quiere decir que *Microsoft* no pueda producir una versión multiplataforma de la tecnología *ActiveX* (de hecho existen rumores fundados sobre este último hecho, incluyendo a las plataformas *UNIX* y *Macintosh)* pero hasta ese posible lanzamiento, los *JavaBeans* tienen todas las de ganar cuando el requerimiento multiplataforma sea una necesidad.

DEPENDENCIA DEL LENGUAJE DE PROGRAMACIÓN

Por otro lado, nos encontramos con el inconveniente de que la tecnología *JavaBeans* es dependiente en el aspecto referido al lenguaje de implementación. En este apartado, conviene indicar que la creación de controles *ActiveX* no es patrimonio exclusivo de *Microsoft,* sino que es posible desarrollarlos con otras herramientas, como por ejemplo *Delphi* o *C++ Builder*. En cualquier caso y debido a la evolución inherente al mundo informático es de esperar que en un futuro no muy lejano otros lenguajes de programación distintos a *Java* ofrezcan la posibilidad de crear *JavaBeans*.

SEGURIDAD

Los controles *ActiveX* no tienen las mismas restricciones de seguridad que los *JavaBeans*. En los controles *ActiveX* una vez que ha sido instalado en el sistema posee el mismo acceso a los recursos que pueda tener cualquier otra aplicación. Podrá acceder, si está progra-

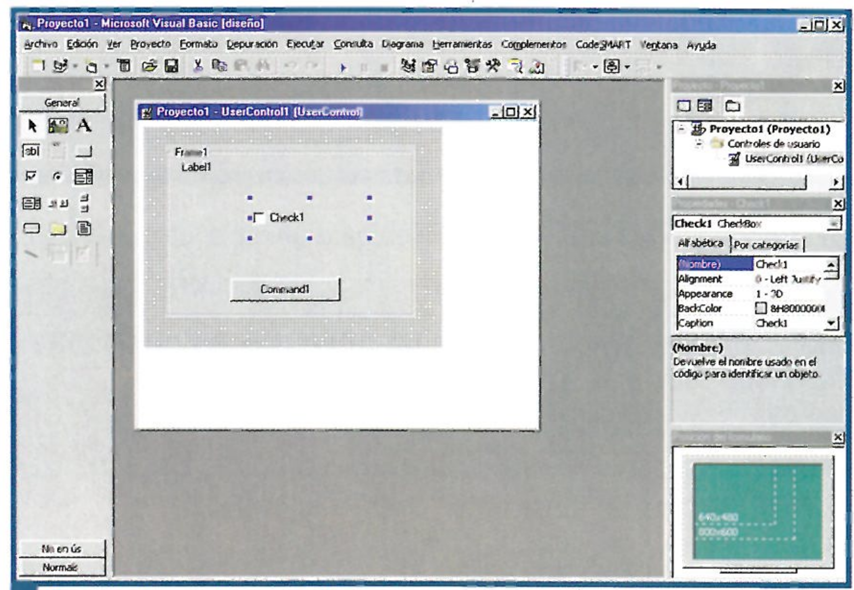
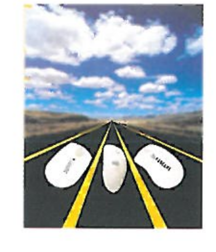


Figura 16.- Proceso de creación de un componente *ActiveX*, en el cual intervienen a su vez diversos componentes *ActiveX*.



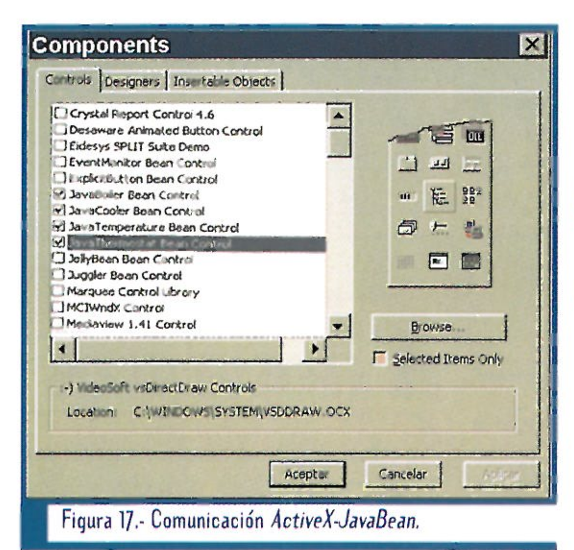


Figura 17.- Comunicación *ActiveX-JavaBean.*

mado para ello, a los ficheros, dispositivos, impresoras, etc. En lugar de restringir las capacidades de los controles *ActiveX,* al contrario que ocurre con las aplicaciones desarrolladas en *Java, Microsoft* utiliza un sistema de firmas de *software* que se ocupa de la seguridad.

La firma es creada con técnicas criptográficas de clave pública

Cada control *ActiveX* descargado a través de *Internet* se encuentra firmado con una firma del desarrollador de *software* que lo ha creado. De forma que si no confiamos en el contenido o en el desarrollador del control podemos desechar su instalación en nuestro sistema.

Esta firma, que es creada con las mismas técnicas criptográficas de clave pública que son utilizadas para crear los certificados seguros de *Web,* permite a los usuarios estar seguros con respecto a las intenciones del programador que ha desarrollado el componente.

Además, para una mayor seguridad, la firma digital *ActiveX* incorpora una tabla *hash* con los contenidos del control, una medida más de seguridad para evitar que el *software* sea modificado.

COMPILACIÓN

Muchos de los lenguajes de *script,* como el incorporado por la tecnología *JavaBeans* no son binarios y no requieren que el código escrito sea compilado y enlazado, lo cual quiere decir que éste también es accesible, al contrario que los controles o la tecnología *ActiveX* donde el código está compilado.

CÓDIGO EXISTENTE

Quizás uno de los puntos más débiles de los *JavaBeans* con respecto a los controles *ActiveX* se refiera al código ya existente para cada opción. *ActiveX* tiene una ventaja significativa, ya que su tecnología *OCX* subyacente es ampliamente usada en la comunidad de *software* de *Windows.* Aunque *Windows* sea únicamente una plataforma, no debemos olvidar que se trata de la plataforma dominante hoy en día en el parqué de ordenadores. Así, los *JavaBeans* tienen que luchar con millones de líneas de código que se encuentran fuertemente basadas en tecnologías directa o indirectamente relacionadas con la tecnología *ActiveX* (como por ejemplo en el caso de *intranets* los paquetes de *Microsoft Office).*

En realidad las dos tecnologías tratadas en este apartado: *JavaBeans* y *ActiveX* tienen cada una por separado ventajas positivas que las convierten en únicas en cierto sentido. Ninguna de las dos se ha mostrado por encima de la otra, y se tiende cada vez más a la integración que permita la convivencia de las dos.

ACTIVEX VERSUS VCL

Enumeremos a continuación los principales rasgos diferenciadores entre estas dos tecnologías de componentes.

- Tamaño del ejecutable.
- Conversión entre tecnologías.
- Lenguaje de programación.
- Encapsulación en contenedores.
- Orientación a objetos.

TAMAÑO DEL EJECUTABLE

Una de las diferencias puede ser el tamaño final del un componente *VCL* respeto a un componente que utilice la tecnología *ActiveX.* Mientras que los componentes *VCL* pueden encapsularse si lo deseamos dentro del ejecutable, los componentes con tecnología *ActiveX* siempre van separados. Esto redunda en un menor tamaño del ejecutable, pero nos obliga a tener *DLL's* (bibliotecas de enlace dinámico) y archivos suplementarios del componente *ActiveX.*

Los componentes VCL, a diferencia de los componentes ActiveX pueden encapsularse dentro del ejecutable

Del mismo modo, si los componentes se encuentran encapsulados dentro del ejecutable, su velocidad de ejecución será superior

ya que en principio, todo el módulo se encuentra en memoria y no es necesario cargar ningún módulo desde disco, con el inconveniente del factor tiempo que esto supone.

CONVERSIÓN ENTRE TECNOLOGÍAS

Una característica muy importante de los componentes de la *VCL* (Biblioteca de componentes visuales) es la posibilidad de convertirlos a otros de tipo *ActiveX*, utilizando el propio entorno de desarrollo. De esta manera, es posible adaptarse a las necesidades concretas de cada aplicación. Hasta la fecha, esta operación no puede efectuarse a la inversa.

LENGUAJE DE PROGRAMACIÓN

Una de las grandes diferencias que encontramos cuando comparamos la tecnología *Active* con otras es la diferencia existente en el número de lenguajes utilizados para crear los componentes reutilizables. Los controles *ActiveX* a diferencia de los componentes *VCL* pueden ser escritos con una amplia variedad de lenguajes, tales como *Visual Basic, C/C++, Delphi, Visual Java,* etc. Es decir, se puede crear un componente de este tipo en el lenguaje que deseemos y utilizarlo dentro del entorno que necesitamos, siempre y cuando soporte la tecnología *ActiveX*.

ENCAPSULACIÓN EN CONTENEDORES

Una de las características de la tecnología *ActiveX* es que va un poco más allá de la creación de componentes (los típicos archivos *OCX* de programación), permitiendo si lo deseamos la creación de *DLLs ActiveX* e incluso ejecutables que sigan la tecnología *ActiveX*.

ORIENTACIÓN A OBJETOS

Centrándonos más en las semejanzas, podemos afirmar que ambas tecnologías soportan la orientación a objetos, así como la manipulación en tiempo de diseño y en tiempo de ejecución. De igual modo, en las dos podemos crear también componentes formados a partir de otros componentes.

COMPARATIVA 2

JAVABEANS VS. ACTIVEX Y VCL

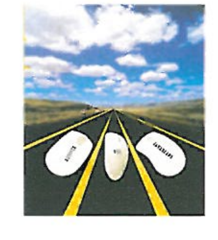
COMPARATIVA DE CARACTERÍSTICAS

Una vez que se han mostrado las características principales de estos tipos de componentes, vamos a realizar una comparativa algo más específica en función de diferentes parámetros, como son el entorno de ejecución, posibilidad de uso de herramientas visuales, lenguaje que utilizar, etc.

La portabilidad es una cualidad muy importante hoy en día. Las grandes redes corporativas incluyen máquinas *Windows,* estilo *Unix,* entornos tradicionales *host,* etc. Los componentes orientados a plataforma *Windows,* como *ActiveX* y *VCL* en este sentido están algo penalizados, y la posibilidad interesante sería el uso de

TABLA 3. Comparativa entre los componentes

	ACTIVEX	JAVABEANS	VCL
Multiplataforma	Entornos Windows	La mayoría de las plataformas	Entornos Windows
Orientación a objetos	Sí	Sí	Sí
Herramientas RAD	Sí	Sí	Sí
Lenguajes	Visual Basic, C, C++ ..	Java	Delphi
Componentes de terceros	Sí	Sí	Sí
Velocidad ejecución	Muy rápido	Rápido	Rápido
Distribución	Sí, DCOM	Sí, EJB	—
Formato	EXE, DLL	JAR, CLASS	EXE
Tamaño componente	Aceptable	Reducido	Grande
Uso aplicaciones Internet	Sí, mediante IIS de Microsoft	Sí, en Java Server Pages, EJB	—
Fabricante relacionado	Microsoft	Sun	Borland



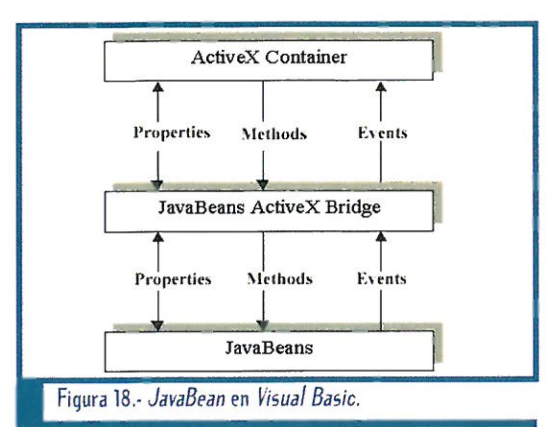


Figura 18.- *JavaBean* en *Visual Basic.*

JavaBeans, que al estar desarrollados en *Java* sí ofrecen una ejecución multiplataforma.

Todos los componentes ofrecen grandes posibilidades con Internet

Esto es algo fundamental en determinadas aplicaciones, y con independencia de posibles problemas de tráfico de red que pudieran ocurrir en el *"download"* de clases, es la opción más probada y utilizada. Respecto a los clásicos componentes *Windows,* tienen a su favor que las especificaciones sobre su creación están "maduras" y no hay dependencias de versiones de *API* como pudiera ocurrir con los *JavaBeans.*

Afortunadamente todos los componentes siguen una filosofía de desarrollo orientado a objetos, por lo que su utilización es sencilla y práctica. Además, gracias a los potentes entornos de desarrollos existentes, comenzar a utilizar un componente requiere poco tiempo y crear uno propio suele pasar por el uso de unos completos y prácticos asistentes que realizan por nosotros la mayoría del trabajo repetitivo y pesado.

Los *ActiveX* suelen generarse usando *C++,* lo que implica un tiempo de aprendizaje algo superior al, por ejemplo, requerido para aprender *Java* y desarrollar *JavaBeans.* A su vez, *Delphi* y *VCL* destacan por requerir un tiempo de aprendizaje mucho menor.

Los componentes *ActiveX* constan de una posibilidad superior al resto, y es que, al menos teóricamente, se pueden construir utilizando múltiples lenguajes, tales como *Visual Basic, C++, Delphi,* etc., que pueden animar a un amplio conjunto de desarrolladores que se han especializado en esos lenguajes.

De todos modos, el resto de componentes pueden convertirse por el uso de diferentes herramientas a *ActiveX,* mediante el uso de por ejemplo *ActiveX-Bridge,* que genera un control *ActiveX* a partir de un componente *JavaBean.*

La razón por la que parece que los *ActiveX* reinan en los entornos gráficos, ya que todo se puede convertir a *ActiveX,* es porque *Windows* es la plataforma de escritorio líder, y para no perder cuota de mercado, el resto de fabricantes *software* siempre presentan su tecnología pero dejando abierta la puerta que permite convertir sus propios componentes al formato del que tiene una mayor implantación, que no por eso tiene que ser la mejor opción.

Todos los tipos de componentes descritos pueden ser utilizados en variadas herramientas visuales, por lo que su facilidad de creación está asegurada. La complejidad del componente está limitada prácticamente a la habilidad del desarrollador y hoy en día existen múltiples fabricantes que venden los denominados componentes de terceros. De esta forma, el desarrollo de una aplicación se asemeja a la construcción de un *puzzle* colocando las piezas creadas por otros. Normalmente estas piezas se encargan de una función muy específica dentro de la problemática global del proyecto abordado.

Con la tecnología ActiveX podemos crear también DLL's y ejecutables ActiveX

Conviene destacar que existe una opinión generalizada incorrecta sobre que los *JavaBeans* no pueden interactuar y manipularse en herramientas de diseño para las que no han sido concebidos. Por lo general nadie espera que un *JavaBean* pueda ser utilizado en *Visual Basic,* lo cual no es cierto. En la figura 18 observamos un ejemplo de cómo se disponen de *beans* para una aplicación *Visual Basic.*

Un aspecto importante en cualquier comparativa se refiere a la velocidad de ejecución de cada tipo de tecnología. Dado que la única forma de poder compararlos es en un entorno *Windows,* los componentes *ActiveX* son los más veloces, con un tamaño reducido, mientras que los *JavaBeans* se muestran algo lentos, aún teniendo un tamaño mucho menor. En el término medio de velocidad aparecen los componentes *VCL,* pero con un tamaño superior a los demás.

Conviene comentar que *Sun* ha dejado a un lado su política restrictiva sobre funcionalidad *Win32* en *Java,* y en las últimas *releases* de *JDK (Java Development Kit)* todo lo relacionado con la parte visual, *beans,*

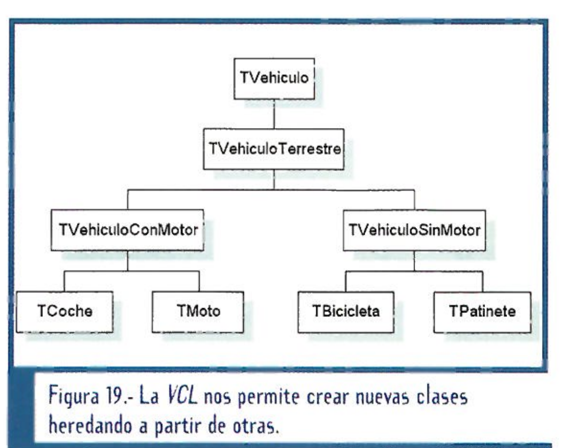


Figura 19.- La *VCL* nos permite crear nuevas clases heredando a partir de otras.

applets, frames, etc., se construye bajo una interfaz nativa *Win32,* disminuyéndose por tanto las diferencias de velocidad de ejecución entre los *ActiveX* y los *JavaBeans.*

Todos los componentes ofrecen amplias posibilidades relacionadas con las comunicaciones, principalmente con *Internet,* por lo que con cualquier tecnología se podrán desarrollar servidores *Web, FTP,* aplicaciones de correo electrónico, etc.

También existen posibilidades relacionadas con distribución, en el caso de *ActiveX* mediante *DCOM* y con los *JavaBeans* mediante los *Enterprise Java Beans (EJB),* pero de todos modos, ninguna de estas dos especificaciones está siendo muy utilizada por su elevada complejidad de desarrollo y por sus altos requerimientos. Además debemos añadir su escasa madurez debido a su novedad.

Por otra parte, destaca la relación de uso entre los *JavaBeans* y los *ActiveX.* Aunque estos componentes no son excluyentes entre sí, al menos teóricamente, y se declara que un *bean* se puede incluir en un *container* de un *ActiveX,* (por supuesto todo esto pasando por el uso *ActiveX-Brigde),* existen ciertos problemas que mencionar para dejar claras sus posibilidades de uso conjunto. Una cosa tan simple como un *bean* invisible presenta ciertas pegas técnicas. Para sorpresa de muchos, *ActiveX-Brigde* no soporta correctamente los *bean* invisibles.

Otra circunstancia que se debe tener en cuenta es la gestión de eventos, ya que no hay un paralelismo de construcción entre el modelo de *JavaBeans* y el de *ActiveX,* por lo que surgen ciertos problemas. Resumiendo, la interrelación entre *ActiveX* y *JavaBeans* no es ni mucho menos inmediata.

En la Figura 17 se muestra la relación entre los *JavaBeans* y los *ActiveX* cuando se utilizan de forma conjunta.

COMPARATIVA 3

VCL Vs. ACTIVEX y JAVABEANS

VCL Vs. ACTIVEX

Tanto la tecnología *VCL* como la *ActiveX* (y por supuesto, los *JavaBeens)* se usan en esencia para lo mismo: reducir los tiempos de desarrollo, reutilizando el mayor código posible.

Gracias a la *VCL,* es posible reducir los tiempos de desarrollo en gran medida. En cuanto detectemos que hay una serie de operaciones que pueden ser encapsuladas, lo más conveniente es crear una estructura de clases que proporcionen a los programadores los métodos que implementen dichas operaciones.

Si además nos interesa interactuar con las clases en tiempo de diseño, podemos crear un componente que las englobe (recordemos que los componentes son parte de la *VCL,* y no la *VCL* completa, como se podría pensar).

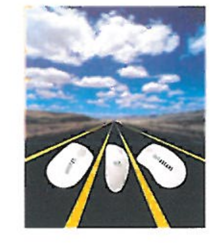
Supongamos que tenemos una serie de funciones, que usamos para representar datos en forma de gráfico. Podríamos agrupar dichas funciones en una única clase, llamada por ejemplo *TGrafico.* Un gráfico tiene muchas propiedades (número de colores, tipo, etc.). Así que decidimos crear un componente que englobe las propiedades, métodos y eventos que pensamos que pueden sernos más útiles.

Las tres tecnologías intentan reducir los tiempos de desarrollo

Una vez creado el componente, lo hacemos de libre distribución (aunque no se incluyan los fuentes). Además, y mediante un asistente, crearemos también un componente *ActiveX* a partir del componente *VCL,* que también distribuiremos.

Algunos días después, recibimos un *E-mail* donde nos preguntan por qué no funciona el *ActiveX* en *Visual Basic.* Muy sencillo, puesto que el *ActiveX* fue creado en *Delphi,* es necesario que en la máquina cliente se encuentren las librerías básicas de *Delphi.* Si el *ActiveX* hubiese sido creado en *C++,* tendríamos que cargar con el *runtime* de dicho lenguaje.

Y éste es uno de los principales problemas de la tecnología *ActiveX.* Un proyecto, donde se usen varios controles *ActiveX,* donde cada uno



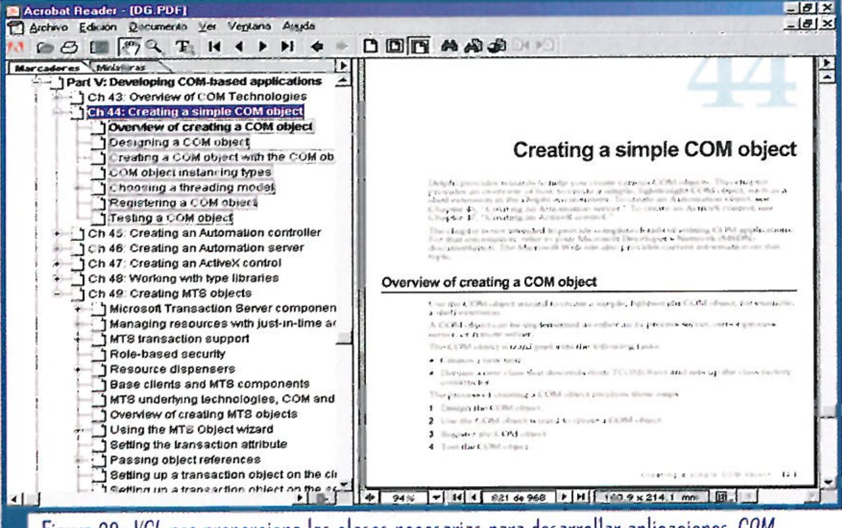


Figura 20- *VCL* nos proporciona las clases necesarias para desarrollar aplicaciones *COM, DCOM, MTS, CORBA,* etcétera.

de ellos proviene de una fuente distinta, y ha sido creado en un lenguaje diferente, tiene que cargar con varios *runtimes,* con el consiguiente gasto de recursos. Esto es debido a que los *ActiveX* se implementan como *DLL's* de *Windows.*

La versión *VCL* se integraría dentro del ejecutable, generándose un código de lo más eficiente, y no habría que cargar con *runtimes* que no hacen más que devorar recursos.

En el mismo correo, nos comentan que sería necesario que el componente *ActiveX* permitiese crear más tipos de gráficos. Si la tecnología *ActiveX* permitiese herencia, el usuario podría crear un nuevo componente, heredando del anterior, al que le habría añadido la capacidad de generar nuevos tipos de gráficos.

Lamentablemente, utilizando la tecnología *ActiveX,* la única posibilidad de ampliar un control consiste en disponer del código fuente (y un lenguaje que lo compile, por supuesto), y hacer los cambios que deseemos. Esto choca un poco con el espíritu de la programación orientada a objetos, y con el concepto de código reutilizable.

Si el componente *TGrafico* funciona bien, ¿qué necesidad hay de modificarlo para incluir nuevas características? Nos arriesgamos a introducir fallos en el componente, debido a los añadidos. ¿No es mucho mejor, crear un nuevo componente, derivado de *TGrafico,* y hacer sobre él todos los cambios?. Esto, tan simple, intuitivo y práctico como parece, no es posible realizarlo con tecnología *ActiveX,* pero sí con la *VCL.*

Ahora, la única posibilidad del usuario es que el programador original cree un nuevo componente *VCL,* que derive del *TGrafico* original, pero con las capacidades ampliadas que el usuario solicita, vuelva a generar un *ActiveX,* y lo vuelva a distribuir.

Por último, en el correo nos comentan que si es posible mejorar la velocidad de ejecución del componente. En efecto, las llamadas a los métodos, eventos y propiedades de los *ActiveX* no son todo lo eficiente que debieran, puesto que todas estas llamadas tienen que pasar por la capa *OLE.* Esto, al final, hace que el componente *ActiveX* no sea tan rápido como su equivalente *VCL.*

Como respuesta a dicho correo, le indicamos al usuario que si lo que desea es reutilizar al máximo el código, y al mismo tiempo, obtener ejecutables rápidos, compactos y que no sean devoradores de recursos, podría utilizar la versión *VCL* del componente. El usuario a su vez nos responde que no cuenta con licencia de *Delphi,* que usa el componente *ActiveX* en varios lenguajes *(Visual C++* y *Visual Basic,* por ejemplo), y que no tiene intenciones de cambiar.

Ésta es, bajo mi punto de vista, la principal desventaja de la *VCL* frente a la tecnología *ActiveX.* Nos vemos obligados a depender de un único lenguaje de programación (o dos, si tenemos en cuenta a *C++ Builder),* tanto para poder usar la *VCL,* como para poder desarrollar componentes.

Con respecto a la compatibilidad entre plataformas, las dos tecnologías están equiparadas. Se limitan al entorno *Windows,* aunque actualmente hay intentos de portar la *VCL* a sistemas *Unix/Linux (Free Pascal,* www.freepascal.org y el futuro *Kylyx,* www.inprise.com)

VCL Vs. JAVABEANS

Sigamos con nuestro ejemplo. Supongamos que tenemos una versión de nuestro componente *TGrafico* implementada como *JavaBean.* Supongamos que queremos representar datos en movimiento (barras que suben y bajan, colores que cambian, etc.). Para ello, vamos variando los datos, y refrescando el gráfico constantemente.

Aquí podemos encontrarnos con el primer problema de los *JavaBeans.* Al estar basados en

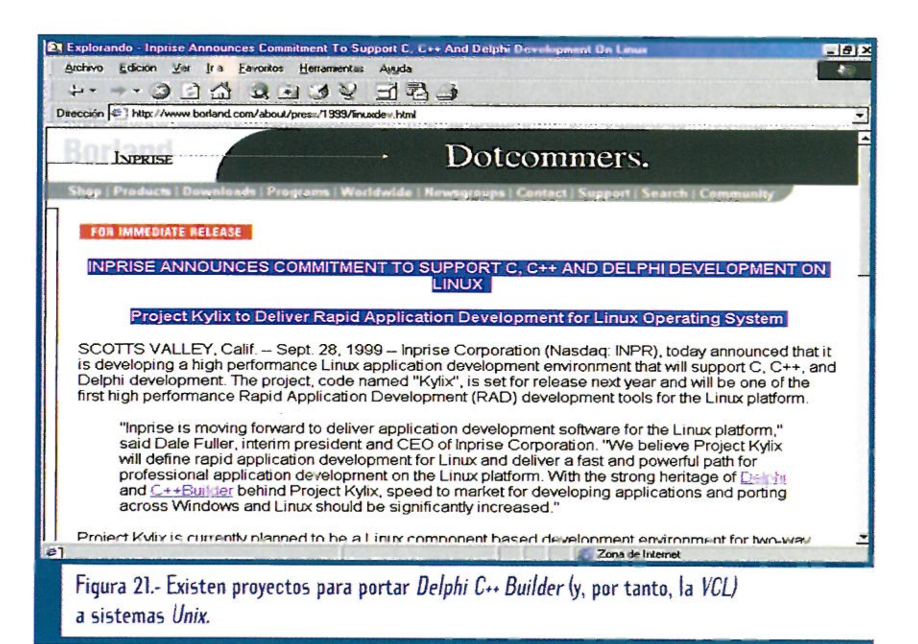


Figura 21.- Existen proyectos para portar *Delphi C++ Builder* (y, por tanto, la *VCL*) a sistemas *Unix*.

Java, son más lentos que sus equivalentes en la *VCL*. Los compiladores de *Java* generan código independiente de la máquina. Es necesario que exista una máquina virtual *Java* en el sistema, que traduzca el código *Java* al código máquina de la *CPU*.

Cuanto más eficiente sea la máquina virtual, mejor se ejecutarán nuestros *Javabeans*. Pero por muy optimizada que esté, al final siempre hay que pasar por más capas de *software,* por lo que su ejecución es más lenta. Además, dependemos de terceras empresas (en este caso, la creadora de la máquina virtual *Java).*

Si ésta falla, o no cumple con los estándares definidos por *Sun,* podemos tener problemas al ejecutar nuestro *JavaBean* en otros sistemas distintos al de desarrollo. Al final, algo que parecía fácil, crear un gráfico en movimiento, no ofrece los resultados que esperábamos, debido precisamente, a la falta de velocidad.

En estos momentos, pensamos que si recurrimos a las funciones de la *API* de *Windows* podemos mejorar la velocidad de ejecución. Pero si lo hacemos, estamos limitando el uso del *JavaBean* al entorno *Windows,* con lo que deja de ser un componente multiplataforma.

Realmente, los componentes *VCL* y los *JavaBeans* son muy parecidos. Tienen una estructura similar, se usan de la misma forma, etc. Las diferencias más apreciables se refieren a la velocidad y la compatibilidad entre plataformas.

La tecnología ActiveX no permite herencia, limitando el concepto de código reutilizable

Los componentes de la *VCL* son muy rápidos y eficientes, gracias a la ventaja que supone el generar código para una única plataforma. Además, permiten acceder a cualquier función de la *API* de *Windows,* así como a los recursos *hardware* de la máquina.

Los *JavaBeans* son más lentos a la hora de ejecutarse, y no permiten aprovechar los recursos que el sistema operativo proporciona (como por ejemplo, el acceso a las diferentes *API's).* Existen compiladores de *Java* que permiten incluir llamadas a la *API* (como los de *Microsoft),* pero en este caso se sacrifica la mayor ventaja de los *JavaBeans,* su portabilidad.

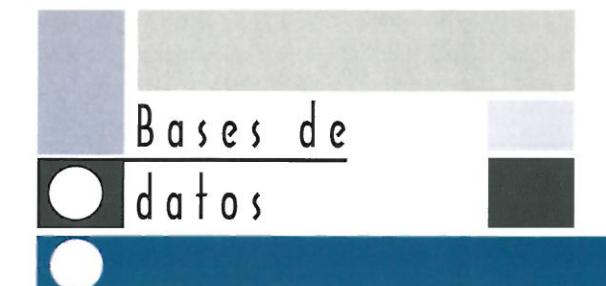
CONCLUSIÓN

Resumiendo podemos decir que no hay ninguna tecnología que se muestre claramente superior a otra. En última instancia, la elección dependerá de nuestras necesidades, o de nuestras preferencias personales a la hora de programar.

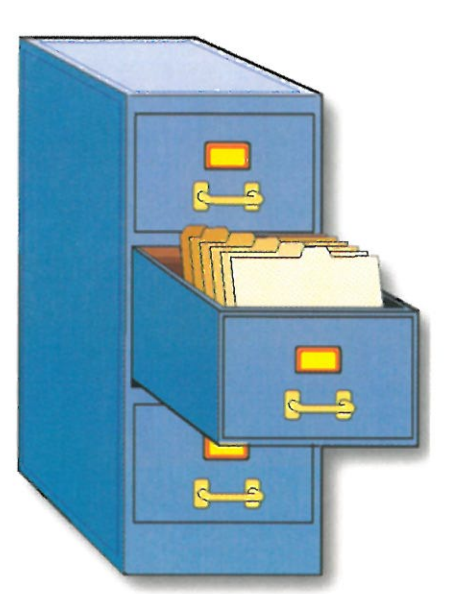
Los componentes JavaBeans suelen ser más lentos que sus equivalentes VCL

De hecho, los diferentes tipos de componentes ofrecen la mayoría de las posibilidades para los desarrolladores, por lo que la decisión sobre el uso de un tipo u otro se puede basar en los requerimientos de la aplicación. El lenguaje puede determinar el tipo de componente que utilizar, aunque el factor que puede inclinar la balanza es, en gran medida, la necesidad o no de portabilidad.

Si en el fondo lo que se necesita es un componente visual, éste no es el que decidirá su uso por sus diferentes posibilidades, sino que la decisión final vendrá tomada por los diferentes requerimientos que se necesiten. De todas formas esperamos que con esta comparativa la elección sea mucho más fácil de tomar a partir de los puntos fuertes de cada tecnología. Desde luego fundamentos a favor de una u otra no van a faltar.



Aplicación de bases de datos con Delphi 5 (II)



Juan Luis Ceada Ramos.
Programador en ARCABE Formación y Servicios Informáticos.

En esta entrega estudiaremos los componentes de acceso y edición de datos. El próximo mes pondremos en práctica todos los conocimientos adquiridos, desarrollando una pequeña aplicación.

BDE: ALIAS

En la entrega anterior hicimos algunos comentarios acerca de los alias. En pocas palabras, un alias es un acceso directo a los datos. Los alias se crean y configuran mediante *BDE Administrator* (y también mediante *SQL Explorer*).

Cuando creamos aplicaciones, de alguna forma hay que indicar dónde estarán situados los datos. También será necesario indicarle el formato de las bases de datos utilizadas, para lo que se utilizan los alias. Cuando una aplicación quiere acceder a un conjunto de datos a través del *BDE*, basta con que le indique qué alias debe usar y a qué tablas desea acceder.

El *BDE* busca en la lista de alias (que se encuentran en el fichero **idapi.cfg**) el que le hemos especificado, y una vez que lo ha encontrado, conoce dónde se encuentran ubicados los datos, y cuál es su formato.

En el caso de las bases de datos de escritorio, el alias apunta al directorio donde están los datos, y poco más. En el caso de

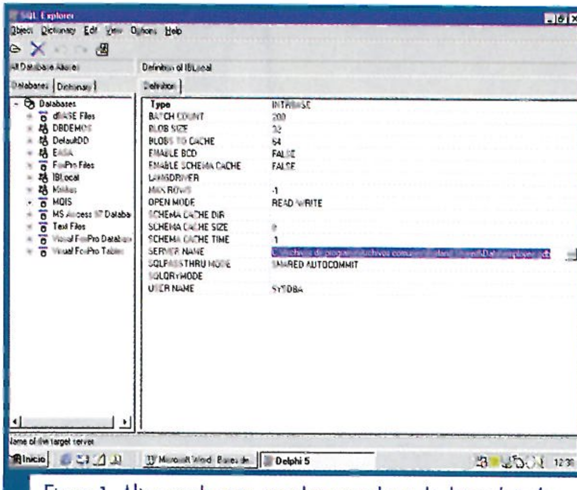


Figura 1.- Alias usado para acceder a una base de datos *Interbase*.

que el alias apunte a una base de datos situada en un servidor *SQL*, el alias lleva asociado una serie de parámetros (ver Figura 1), que en la mayoría de los casos no será necesario modificar (exceptuando, claro está, el parámetro que indica a qué base de datos queremos acceder).

El uso de alias presenta dos ventajas principales:

Independencia de la ubicación de los datos: es posible desarrollar una aplicación en casa que acceda a datos situados en un disco duro local, y llevarla a nuestra empresa, donde los datos se encuentran en una unidad de red. La aplicación seguirá funcionando sin necesidad de ser recompilada. Basta con que exista un alias en ambos ordenadores (el de casa y el del trabajo) con el mismo nombre, y que apunte a los datos.

La clase TDataSet proporciona los métodos básicos para poder trabajar con los datos

Independencia del formato de los datos: supongamos que en casa los datos los tenemos en tablas *Paradox*, pero en la empresa trabajamos con un servidor. Tampoco hay problema, basta con que el alias del ordenador del trabajo se llame igual que el de casa, pero que apunte a la base de datos en el servidor *Oracle*. Nuestra aplicación funcionará en ambos entornos (doméstico y profesional), sin tener que recompilar.

Existen dos tipos de alias, los persistentes y los locales. Los primeros son los que se crean usando el *BDE Administrator*, y están disponibles para cualquier aplicación que esté instalada en la máquina. Los segundos sólo existen durante la ejecución de una determinada aplicación, y se crean usando componentes *TDatabase* (hablaremos de este componente en próximas entregas).

Por último, basta indicar que para los ejemplos que veamos a lo largo de este artículo, usaremos el alias **DBDEMOS**. Dicho alias se crea durante la instalación de *Delphi*, siempre y cuando hayamos instalado las aplicaciones y datos de ejemplo. Normalmente apunta al directorio **C:\Archivos de programa\Archivos comunes\Borland Shared\Data**, y utiliza un formato de datos *Paradox*.

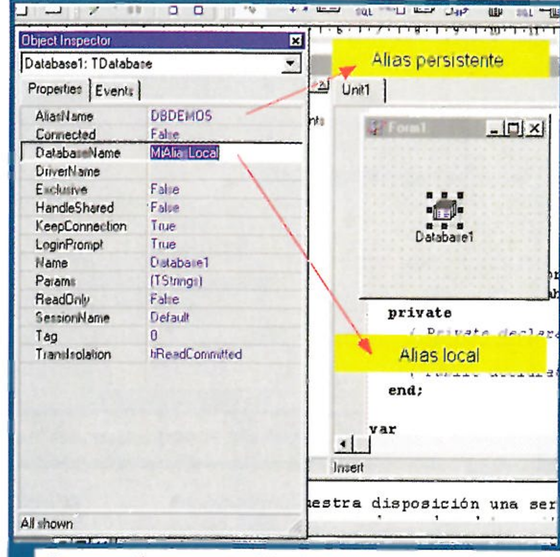


Figura 2.- El componente *TDatabase* permite crear alias locales.

ACCESO A LOS DATOS

Delphi pone a nuestra disposición una serie de componentes para poder acceder a los datos situados en las bases de datos. Dichos controles se encuentran en la paleta *Data Access*.

CLASE TDATASET

Es la antecesora de las clases *TTable* y *TQuery*, y define las operaciones básicas para acceder y navegar por los datos. Además, gracias a ella podemos conocer el estado de un conjunto de datos y realizar el control de errores. En ella se declaran los métodos y eventos que son comunes a las clases hijas.

Esta clase proporciona uniformidad. Por ejemplo, usaremos el mismo método (*First*), si queremos ir al inicio de un conjunto de datos, da igual el componente que hayamos usado para acceder a ellos (un *TTable* o un *TQuery*).

La clase TDataSet define los métodos, eventos y propiedades comunes a TTable y TQuery

En la mayoría de los casos no trabajaremos con esta clase directamente, sino con sus descendientes, *TTable* y *TQuery*. Veamos ahora qué métodos y eventos proporciona esta clase a sus hijas.

MÉTODOS DE TDATASET

A continuación se describen, de manera esquemática, los principales métodos de la clase *TDataSet*. Conviene familiarizarse con ellos, puesto que en posteriores entregas haremos uso de los más relevantes.

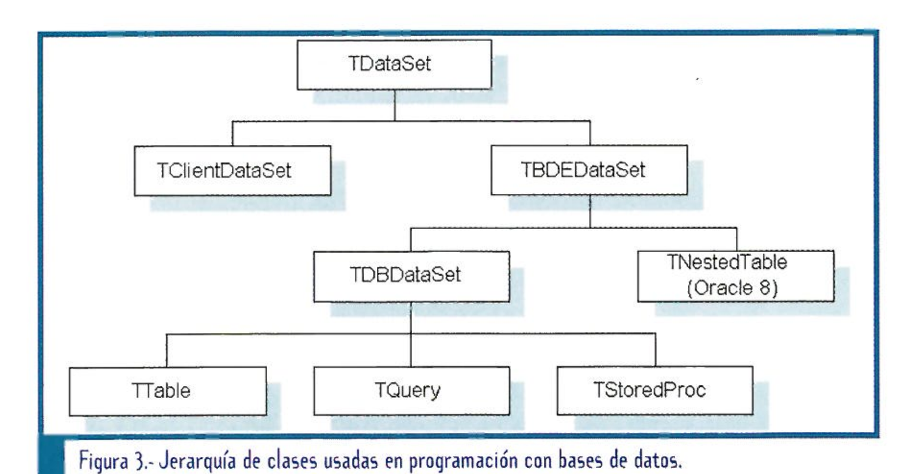


Figura 3.- Jerarquía de clases usadas en programación con bases de datos.

- *Append*: inserta un registro en la tabla asociada al *dataset*. El estado del *dataset* (ya sea tabla o consulta) pasa a Inserción.
- *Cancel*: cancela los cambios realizados desde que la última operación *edit* o *append*. El estado del *dataset* pasa a Navegación.
- *Open*, Close: abre/cierra el *dataset.*
- *Edit*: editamos el registro actual. El estado del *dataset* pasa a Edición.
- *Locate*, *FindNext*, *FindLast*, *FindFirst*, *FindPrior*: permiten desplazarse y buscar a lo largo del conjunto de datos. Los veremos con más detalle en el próximo artículo.
- *First*, *Last*, *Prior*, *Next*: el cursor se desplaza al principio, final, registro anterior o posterior respectivamente. Si la tabla está en modo de edición/inserción, realizan un *Post*, y ponen la tabla en modo Navegación.
- *Insert*: inserta un nuevo registro en la posición actual. El *dataset* pasa a modo de Inserción.
- *IsEmpty*: función que devuelve un valor *booleano*, que indica si el conjunto de datos está vacío.
- *Refresh*: actualiza el conjunto de datos, lo que permite, por ejemplo, ver los cambios realizados por otros usuarios. Si el *dataset* está en modo de edición/inserción, se realiza un *Post*, pasando al modo de Navegación.
- *Post*: acepta todos los cambios que hayamos realizado en el registro actual, almacenándolos en la base de datos. Pasa del modo edición/inserción al modo de Navegación.

EVENTOS DE TDATASET

La mayoría de los eventos que proporciona esta clase son del tipo *Before/After*. Por ejemplo, tenemos un evento *BeforeCancel* que se lanza inmediatamente antes de cancelar la operación actual (y que permite que anulemos dicha operación, simplemente lanzando una excepción). Y tenemos un evento *AfterCancel*, que se lanza justo después de finalizar la cancelación de los datos.

Así, tenemos un par de eventos de este tipo para cada uno de los métodos más importantes: *Before /AfterClose, Before/AfterDelete, Before /AfterEdit, Before/AfterInsert, Before /AfterOpen, Before/AfterPost, Before /AfterRefresh, Before/AfterScroll* (estos dos se producen cuando navegamos por el conjunto de datos).

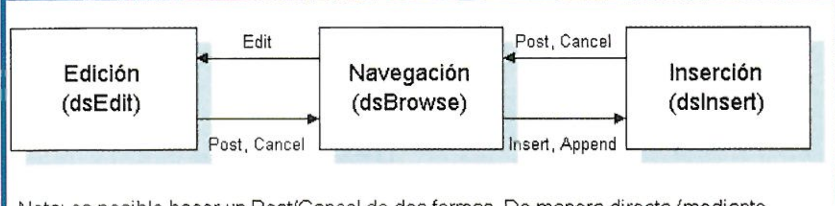
Hay dos tipos de alias: persistentes y locales

Además, existen algunos eventos más, como *OnDeleteError*, que salta cuando se produce un error al borrar los datos, *OnEditError* y *OnPostError*. Por último, el evento *OnNewRecord* se produce cuando se introduce un nuevo registro en la tabla, y permite, por ejemplo, asignar valores iniciales a determinados campos de la tabla o consulta.

ESTADOS DE UN DATASET

Un *dataset* (recordemos, una tabla o consulta), puede estar en varios estados. Podemos consultar el estado en el que se encuentra mediante la propiedad *State* (por ejemplo, *Tabla.State in [dsEdit]*). El *dataset* va cambiando de estado en función de los métodos que ejecutemos. Existen tres estados principales: Inserción (*dsInsert*), Edición (*dsEdit*) y Navegación (*dsBrowse*).

Entramos en estado de Inserción cuando realizamos un *Append*



Nota: es posible hacer un Post/Cancel de dos formas. De manera directa (mediante los métodos Post y Cancel), o indirecta (mediante los métodos Next, Prior, Refresh, etc....)

Figura 4.- Estados de un *Dataset* (sólo se muestran los principales).



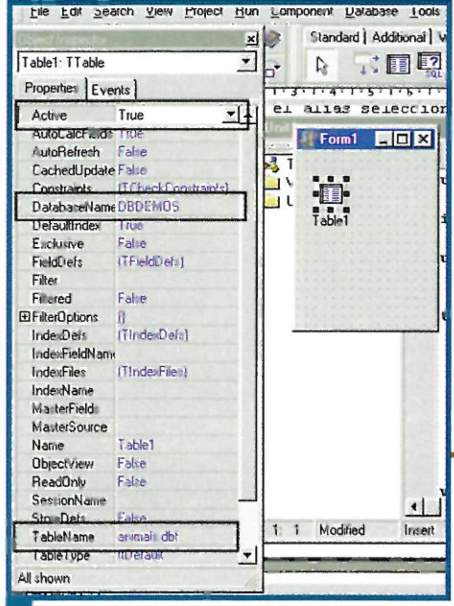


Figura 5.- Componente *TTable* "enganchado" a la tabla animals.db.

o un *Insert*. Pasamos a modo de Edición después de llamar al método *Edit*. Por último, pasamos a modo de Navegación cuando realizamos un *Post* o un *Cancel*. Hay veces que sin llamar a ninguno de estos dos métodos, el *dataset* pasa al modo de Navegación de forma automática.

En general, cualquier método que pueda provocar un movimiento en el conjunto de datos (por ejemplo, el método *Next*), realiza un *post* automáticamente si el *dataset* se encuentra en modo de edición/inserción.

COMPONENTE TTABLE

El componente *TTable* representa a tablas, tanto físicas como "virtuales" (por ejemplo, una vista de datos), que pertenecen a una base de datos.

El componente TTable permite acceder a tablas físicas o virtuales

En la Tabla 1 se muestran las principales propiedades del componente *TTable*. Para configurar este componente, hay que seguir los siguientes pasos:

- Establecer en la propiedad *DatabaseName* el alias (local o persistente) que usaremos (por ejemplo, **DBDEMOS**).

Un dataset puede estar en tres estados principales: Edición, Inserción y Navegación

- Indicar la tabla que queremos abrir (mediante la propiedad *TableName*). El *combo* que se despliega al hacer *clic* sobre esta propiedad en el *Inspector de Objetos* muestra todas las tablas a las que podemos acceder usando el alias seleccionado (por ejemplo, **animals.db**).
- Cambiar el valor de la propiedad *Active* a **True**.
- En estos momentos, nuestro componente *TTable* se encuentra conectado a la tabla **animals.db.** Todos los componentes de edición mostrarán los datos del primer registro de la tabla (o de varios registros si se trata de un *TDBgrid*). Inicialmente, la tabla se encontrará en modo *dsBrowse*.

TABLA 1. Principales propiedades de TTable.

Propiedad	Descripción
Active	Si vale ***True*** la tabla se abre. Si vale ***Flase*** se cierra.
DataBaseName	Indica el alias que usaremos para acceder a los datos.
Exclusive	Indica si la tabla se abre en modo exclusivo (no permite accesos concurrentes). Útil en algunos casos.
Filter	Permite especificar un filtro a los datos. Por ejemplo NOMBRE = 'PEPE', sólo se muestran registros con ese nombre.
Filtered	Activa o desactiva el filtro.
IndexName	Permite especificar el índice a usar por el TTable (índices primarios, secundarios, etc.)
MasterField	Indica qué campos se utilizan para enlazar los datasources en relaciones master/detail. Lo veremos en el próximo artículo.
MasterSource	Indica qué Datasource realiza las funciones de maestro en una relación master/detail.
ReadOnly	Especifica si la tabla se abre en modo lectura.
TableName	Indica el nombre de la tabla a abrir.
SessionName	Asocia el TTable a un componente TSession.

COMPONENTE TQUERY

El componente *TQuery* representa a consultas *SQL*. Estas consultas pueden hacer referencia a una o varias tablas de la base de datos. Las consultas *SQL* se pueden realizar sobre servidores *SQL* (en cuyo caso el servidor es el encargado de procesar la consulta), o sobre bases

de datos de escritorio (en este caso, es el *BDE* el encargado de procesarlas). En la Tabla 2 se muestran sus principales propiedades.

El componente TQuery es útil en cualquier sistema, no sólo al trabajar con servidores SQL

Aunque se pueda pensar que un componente *TQuery* queda relegado a sistemas donde se trabaje con un servidor de datos *SQL*, nada más lejos de la realidad. Este componente es útil en cualquier caso. En sistemas basados en bases de datos de escritorio pueden usarse, por ejemplo, para obtener datos combinados de varias tablas, de una forma rápida y sencilla (como por ejemplo para sacar un listado de estadísticas).

En este caso, la configuración es algo diferente. Primero tendremos que seleccionar el alias que usar mediante la propiedad *DataBase Name*. Después, insertaremos la sentencia *SQL* que deseemos lanzar en la propiedad *SQL*. Por último, lanzaremos la consulta, poniendo la propiedad *Active* a *True* (o en su defecto, usaremos los métodos *Open* y *Close*).

COMPONENTE TDATASOURCE

Un conjunto de datos, ya sea tabla o consulta, no puede visualizar los datos con los que trabaja. Para comunicarse con los controles de edición, se necesita usar un componente auxiliar: *TDataSource*. Dicho componente se conecta a la tabla o consulta adecuada (asignándosela a su propiedad *Dataset*).

TABLA 2. Principales propiedades de TQuery

Propiedad	Descripción
Active	Ver Tabla 1
DataBaseName	Ver Tabla 1
Filter	Ver Tabla 1
Filtered	Ver Tabla 1
IndexName	Ver Tabla 1
MasterField	Ver Tabla 1
MasterSource	Ver Tabla 1
Params	Es posible pasar parámetros a una consulta SQL, que nos permiten obtener unos resultados u otros.
RequestLive	Indica si los datos devueltos serán sólo de visualización, o pueden ser editados. Sólo en determinados casos se pueden editar los datos que devuelve una consulta.
SessionName	Asocia el TQuery a un componente TSession.
SQL	Indica la consulta a ejecutar. Puede modificarse en tiempo de diseño y ejecución.

A su vez, los componentes de visualización/edición de datos se conectan al *TDataSource*. Es posible conectar varios *TDataSource* a una misma tabla o consulta. Otra propiedad de este componente es *AutoEdit*. Cuando está a **True**, no es necesario pasar al modo de edición/inserción de forma manual (es decir, efectuando un *Append* o *Edit*). En cuanto el usuario modifica alguno de los datos, automáticamente se pasa al modo de Edición/Inserción.

Todos los componentes que conectemos a un *TDatasource* serán notificados de los cambios de estado y contenido del conjunto de datos. Además, gracias a los eventos *OnDataChange*, *OnStateChange* y *OnUpdateChange*, podemos saber en todo momento cuándo se produce alguna variación, cuándo cambia

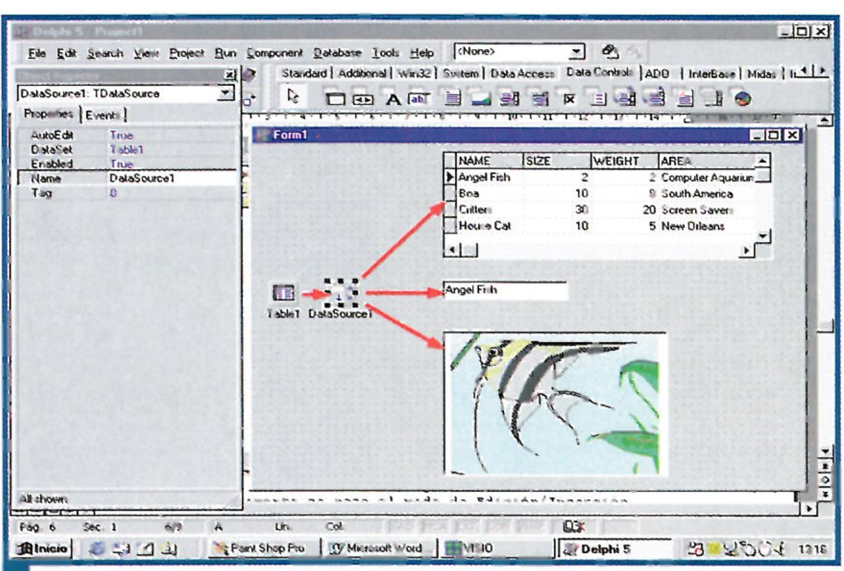


Figura 6.- Es posible visualizar/editar los datos gracias al componente *TDatasource*.



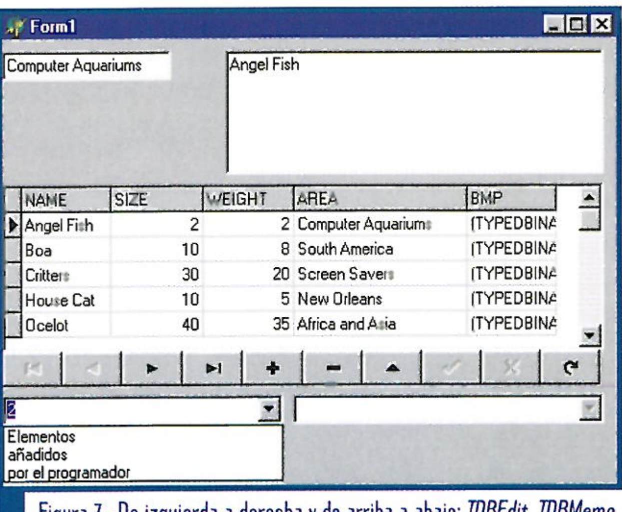


Figura 7.- De izquierda a derecha y de arriba a abajo: *TDBEdit, TDBMemo, TDBMemo, TDBGrid, TDBNavigator, TDBComboBox, TDBLookUpComboBox.*

el estado y en qué momento se actualizan los cambios en el conjunto de datos, respectivamente.

EDICIÓN Y VISUALIZACIÓN DE DATOS

Para poder ver o editar los datos, necesitamos los componentes que se encuentran en la pestaña *Data Controls*. Aunque hay muchos componentes distintos, aquí sólo vamos a ver los principales. El resto son muy parecidos a los que aquí se exponen, por lo que el lector deberá dedicar algo de tiempo a investigar su funcionamiento.

TDataset proporciona eventos que se lanzan antes y después de una operación

En cuanto conectemos los componentes de forma adecuada, y activemos la tabla o consulta, se mostrarán los datos en estos controles. En cualquier momento el usuario puede posicionarse en ellos y comenzar a introducir datos. Si *TDataSource.AutoEdit* es *True*, la tabla pasará a modo de Edición en cuanto el usuario teclee. En otro caso, aunque el usuario teclee, no se apreciará ningún cambio, a menos que se haya llamado al método *Edit* o *Append*.

COMPONENTES TDBEDIT Y TDBMEMO

Es el equivalente a los controles *TEdit* y *TMemo*, pero para bases de datos. Por ello nos limitaremos a comentar las nuevas propiedades:

- *DataSource*: indica el componente *TDataSource* al que lo enlazamos
- *DataField*: indica el campo que se va a visualizar en el control. Depende directamente de la propiedad *Data Source* (hasta que no la asignemos, no podremos seleccionar el campo a visualizar).

COMPONENTE TDBGRID

Permite visualizar/modificar varios registros a la vez de una determinada tabla. El componente es equivalente a *TStringGrid*, es decir, una rejilla de datos, donde el usuario (si el programador se lo permite) puede moverse hacia arriba y abajo, a la izquierda y a la derecha, mover y redimensionar columnas, etc.

Para visualizar los datos es necesario enlazar un TDatasource a cada dataset

La propiedad más novedosa es, como no, *DataSource*, que enlaza con el componente *TDatasource*, que a su vez enlaza con los datos. En la mayoría de los casos a este componente le acompaña un *TDBNavigator*, que

Species No	Category	Common_Name	Species Name
90020	Triggerfish	Clown Triggerfish	Ballistoides conspicillum
90030	Snapper	Red Emperor	Lutjanus sebae
90050	Wrasse	Giant Maori Wrasse	Cheilinus undulatus
90070	Angelfish	Blue Angelfish	Pomacanthus nauarchus
90080	Cod	Lunartail Rockcod	Variola louti
90090	Scorpionfish	Firefish	Pterois volitans
90100	Butterflyfish	Ornate Butterflyfish	Chaetodon Ornatissimus
90110	Shark	Swell Shark	Cephaloscyllium ventriosun
90120	Ray	Bat Ray	Myliobatis californica
90130	Eel	California Moray	Gymnothorax mordax

Figura 8.- Dos *TDB Navigator* asociados a un mismo *TDBGrid.*

actúa como barra de acceso rápida a acciones como insertar, actualizar, borrar, etc.

COMPONENTE TDBNAVIGATOR

Permite realizar, mediante *clics* del ratón las principales operaciones que se pueden efectuar en un conjunto de datos: añadir, borrar, editar, refrescar y moverse por los datos.

Existen múltiples controles que permiten editar y visualizar los datos

El programador puede decidir qué botones mostrar y cuáles no. Incluso, mediante el evento *BeforeAction*, es posible interceptar qué botón ha pulsado el usuario, y si es necesario, podemos abortar la operación, o efectuar cualquier otra acción asociada.

Por supuesto, como todos los controles, tiene una propiedad *DataSource* a la que hay que asignar algún *TDataSource*.

COMPONENTES TIPO COMBO

Básicamente, existen dos tipos. El primero se trata de un *TComboBox* al que se le ha añadido, al igual que al *TDBEdit*, las propiedades *DataField* y *Data Source*. El programador, en tiempo de diseño añade elementos a su propiedad *Items*. En tiempo de ejecución, el usuario selecciona uno de los elementos, que automáticamente se asigna al campo que hemos indicado en la propiedad *DataField*. Por supuesto, también es posible añadir nuevos elementos al *combo* en tiempo de ejecución, pudiendo proceder de una consulta *SQL*, por ejemplo.

El segundo tipo es parecido al anterior, pero esta vez los elementos que el usuario puede seleccionar se toman de una segunda tabla o consulta. Para ello, usaremos un *TDBLookUpComboBox*, al que asignaremos los siguientes valores:

- *DataSource*: el *TDatasource* que está enlazado con la tabla a modificar (por ejemplo, **DS_Pedido**).
- *Datafield*: el campo que se va a modificar (por ejemplo *COD_CLIENTE*).
- ListSource: *TDatasource* de la tabla/consulta de la que se extraen los datos (por ejemplo, **DS_ListaClientes**).
- *KeyField*: campo de la tabla/consulta de la que se extraen los datos que se asignan al campo indicado en *DataField* (por ejemplo, *CODIGO*, se asigna a *COD_CLIENTE*).
- *ListField*: campo que se muestra en el *combo* (por ejemplo, *NOMBRE CLIENTE*).

OTROS COMPONENTES

Debido a la gran cantidad de componentes de acceso a datos, se deja al lector el estudio de otros componentes, como por ejemplo *TDBText* (para mostrar datos, sin posibilidad de editarlos), *TDBImage* (para mostrar imágenes), *TDBListBox*, *TDBCheckBox*, *TDBRadioGroup*, *TDBRichEdit* (edición de texto formateado), *TDBCtrlGrid* (control de rejilla avanzado) y *TDBChart* (para realizar gráficas a partir de los datos, ver Figura 9).

En el siguiente artículo de la serie realizaremos una pequeña aplicación donde mostraremos cómo usar todos estos componentes. Además, veremos qué son las relaciones *master/detail* y cómo se implementan. También abordaremos la forma de acceder a los campos de un *dataset* en tiempo de ejecución, desde el código de nuestra aplicación. De esta manera, podremos poner en práctica todos los conceptos explicados hasta este momento.

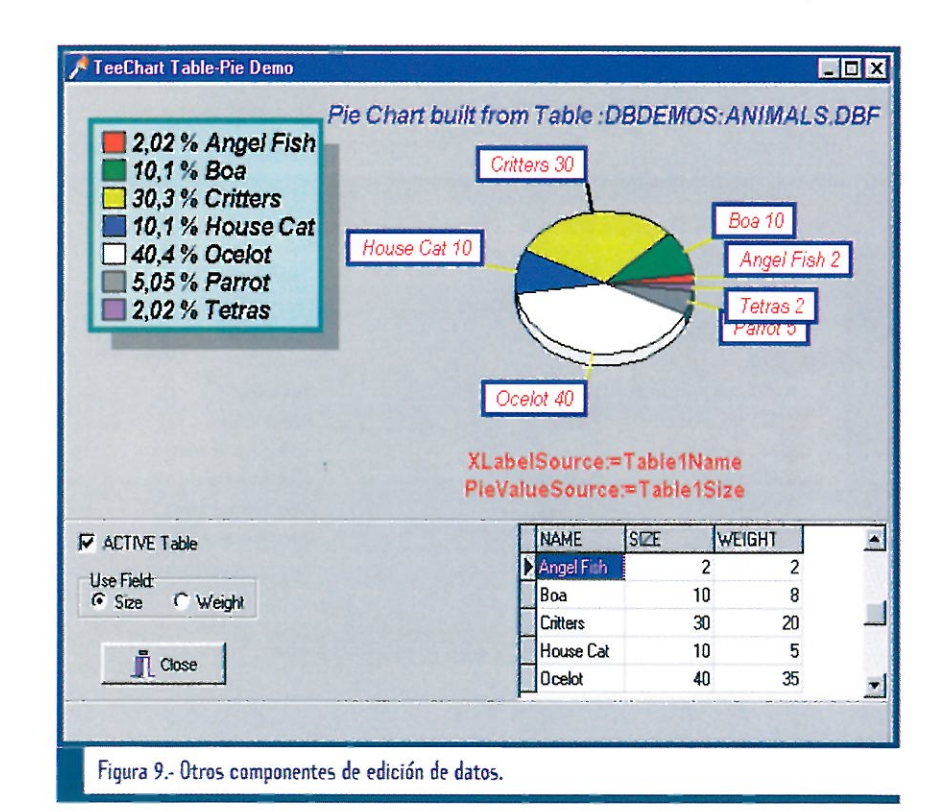


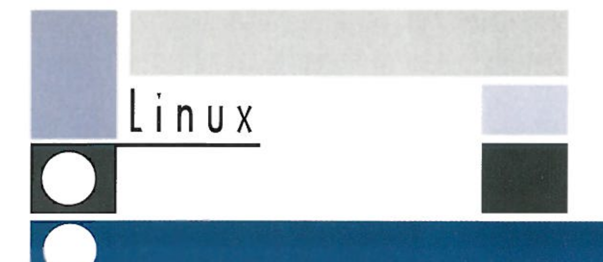
Figura 9.- Otros componentes de edición de datos.

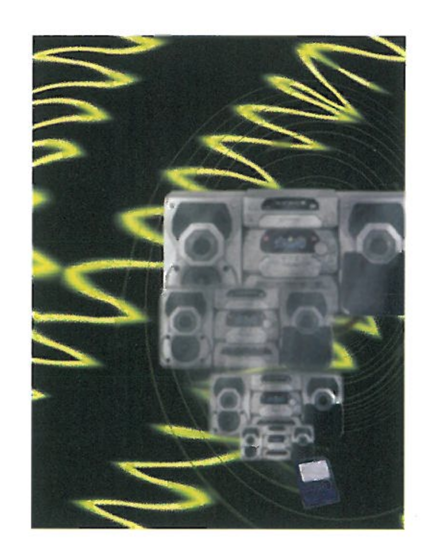
TODOS LOS MESES EN TU QUIOSCO



PARA ESTAR AL DÍA
EN SOFTWARE
SIN GASTARSE
UN EURO EN TELÉFONO

- LOS MEJORES ESPECIALES
- DISEÑO
- INTERNET
- MULTIMEDIA
- UTILIDADES
- EN CASTELLANO
- NEGOCIOS
- LO ÚLTIMO
- Y MUCHO MAS...





MP3 (I): Introducción y características

Vicente A. Sánchez Werner.
Desarrollador Independiente.

En la actualidad, el formato *MP3* se ha convertido en el medio más popular para almacenar información musical gracias a su reducido espacio, a la alta calidad de reproducción y el pequeño coste de descompresión que requiere. Estas propiedades le han convertido en el rey de *Internet,* desafiando incluso el poder de las poderosas compañías discográficas.

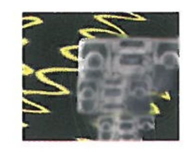
INTRODUCCIÓN

La tecnología que hay detrás de los *MP3* ha llegado rodeada de una gran polémica, alimentada por gran cantidad de intereses económicos. Sin embargo, lejos de reducir su implantación, ha provocado una difusión aún mayor que no ha tenido repercusiones en las ventas de los medios tradicionales de difusión de la música.

La tecnología *MP3* es una tecnología desarrollada por el grupo *MPEG* para su utilización como estándar de audio en los estándares de vídeo *MPEG-1* y *MPEG-2*. Algunos lectores ya sabrán que esta tecnología define formatos de compresión y de reproducción para medios audiovisuales bajo determinadas condiciones de velocidad (ancho de banda), calidad y complejidad. El desarrollo en conjunción con dichos estándares ha llevado a una amplia difusión del formato, en parte gracias a que al ser estándares se han tenido que divulgar implementaciones de referencia, poco eficientes, pero que han servido de base a posteriores desarrollos independientes, especialmente en el mundo *Linux*.

¿QUÉ ES EL MP3?

Existen diferentes respuestas a esta pregunta, ya que algunas personas dirían que se trata de un formato de fichero, otras que es un sistema de compresión y otras que es un formato de audio digital. En cualquier caso todas estas afirmaciones tienen parte de razón, ya que el término está relacionado con todas ellas. Exactamente podríamos decir que *MP3* (abreviatura de *MPEG Layer-3)* es un sistema de compresión definido por la organi-



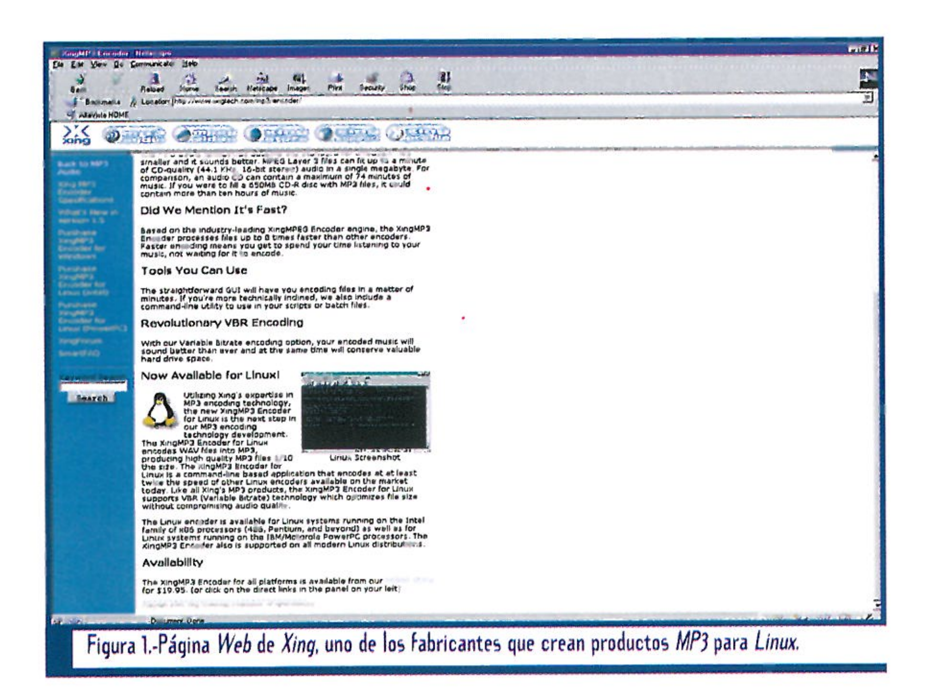


Figura 1.-Página *Web* de *Xing*, uno de los fabricantes que crean productos *MP3* para *Linux*.

zación *MPEG* para la compresión de información sonora. Este término no define exactamente un formato, pese a que existe un estándar publicado por la *ISO* (Organización Internacional de Estándares), ya que puede incluirse en el interior de otros formatos, por ejemplo *Apple QuickTime.*

El conocer qué es el *MP3* abre las puertas a otros interrogantes y requiere definiciones acerca de los términos que hemos usado. El conocimiento más profundo permitirá hacernos una idea más clara sobre lo que podemos esperar del sistema y cómo podemos utilizarlo en nuestros propios proyectos.

Más que un formato de archivo, el concepto de MP3 se acerca más a un conjunto de reglas estandarizadas

A lo largo de la explicación anterior hemos hablado acerca del grupo *MPEG,* pero no hemos dicho qué o quiénes son y cuál es su función. Este grupo se llama *Motion Photographs Experts Group* o grupo de expertos en fotografías en movimiento (animaciones, secuencias de vídeo, etc.) y tiene como función la creación de estándares de vídeo y multimedia que se adapten a las necesidades de la industria. Pese a ser, en teoría, una organización especializada en las imágenes, han creado diversos estándares de audio, ya que éste suele ir íntimamente ligado a las imágenes en movimiento.

Actualmente hay dos estándares en uso *(MPEG-1* y *MPEG-2)* y otros dos *(MPEG-4* y *MPEG-7)* que aún están en fase de definición y desarrollo (aunque el *MPEG-4* está casi terminado y ya existen implementaciones de referencia del mismo). Estos formatos definen cómo deben tratarse las imágenes, el sonido, y cómo deben ser comprimidas y organizadas en el interior de los archivos (cabe destacar que el *MPEG-4* usará el formato de los archivos *QuickTime).*

En la definición de los estándares *MPEG-1* y *MPEG-2* se optó por incorporar la compresión del sonido siguiendo un modelo de capas, cada una de las cuales era más compleja que las anteriores, a la vez que podía reproducir la información generada por las anteriores. Cada una de estas capas se denominó *layer.*

En la actualidad los dos sistemas en uso ya mencionados utilizan el mismo modelo de compresión y disponen del mismo número de capas, que en este caso son tres. Estas capas indican el nivel de complejidad del algoritmo utilizado y dan una idea del ancho de banda que requieren para representar el sonido con una cierta calidad. En concreto la capa 1 es la más sencilla, y la que mayores requerimientos de ancho de banda presenta, mientras que la tres es mucho más compleja, pero requiere mucho menos ancho de banda. De esta forma disponemos de tres capas, cada una adecuada a la reproducción de archivos en situaciones muy determinadas.

En cualquier caso, la capa que nos interesa es la tercera, conocida popularmente como *MP3*. El motivo por el que el término da lugar a confusión es básicamente la creación de la extensión *.MP3* para identificar a estos archivos y su amplio uso en *Internet.*

TECNOLOGÍAS TRAS EL MP3

Este modelo de compresión está respaldado por una nueva metodología de compresión denominada Codificación Perceptual. Esta tecnología permite codificar la información musical de forma altamente compacta, al precio de eliminar algunos rasgos de la misma (de forma similar a como

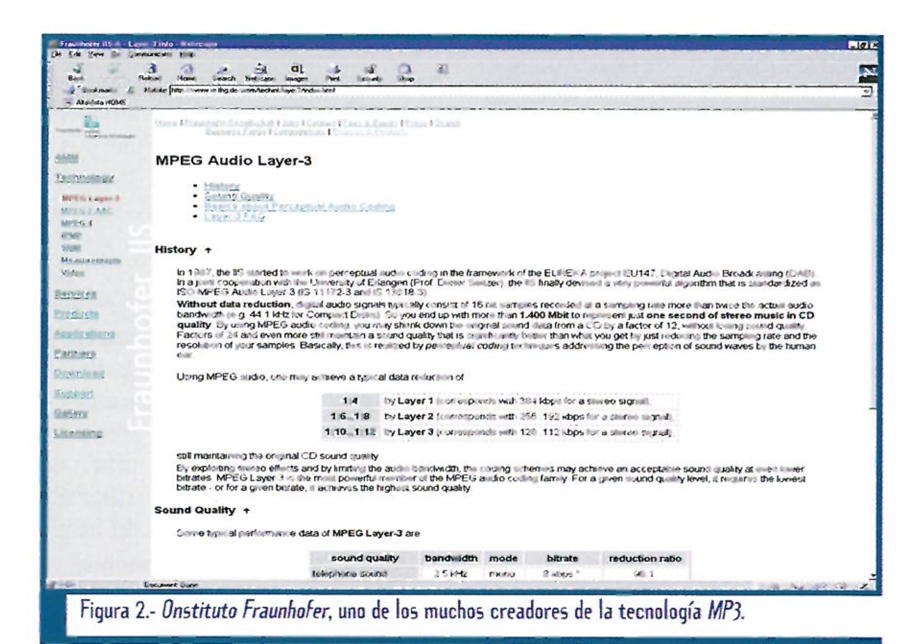


Figura 2.- *Onstituto Fraunhofer*, uno de los muchos creadores de la tecnología *MP3*.

opera el formato de compresión de imágenes *JPEG)*.

Esta técnica de compresión se basa en la percepción del sonido que recibimos a través de nuestros oídos. En condiciones normales y en personas que no hayan educado su oído, éste no percibe determinados matices del sonido o de la música, bien porque queden enmascarados por un sonido de mayor volumen o porque en esa zona confluyan muchos sonidos similares. Es en esa situación en donde entra en juego esta técnica, eliminando el sonido que en teoría no va a ser percibido.

El MP3 es en realidad la tercera capa de compresión de audio definida en los estándares MPEG-1 y MPEG-2

Es por ello que no puede decirse que el *MP3* posea la misma calidad que un *CD*, ya que en este último, no se elimina información, pero también es cierto que la pérdida resulta inapreciable. La sofisticación de esta técnica requiere disponer de máquinas potentes para realizar la compresión en tiempos aceptables, pero la descompresión de la información puede realizarse en tiempo real en equipos de bajas prestaciones.

Esta sofisticada tecnología permite transformar un tema musical de 5 minutos de duración, que ocuparía 50,5 Mb, en un archivo de escasos 5 Mb con una calidad de reproducción prácticamente indistinguible de un *CD*, siempre y cuando no fuese a ser usado en equipos de alta fidelidad (se apreciarían entonces algunas diferencias). Esta reducción de espacio permite la incorporación de sonidos y música de calidad en medios en los que antes era muy difícil de conseguir o la calidad era deficiente como *Internet* o en los *CD-ROM* multimedia.

Antes de seguir y adentrarnos en las características que nos van a permitir saber cómo debemos crear los archivos *MP3* para que resulten útiles para nuestros proyectos, debemos comentar que el *MP3* no es la actual vanguardia tecnológica en el campo de la compresión del sonido, y que existen formatos y modelos que rivalizan en prestaciones o mejoran sus posibilidades, en diversos campos y situaciones.

EL formato MP3 realiza una compresión con pérdida, pero esta diferencia no es apreciable en equipos reproductores como un ordenador

Entre estas tecnologías alternativas podemos encontrar el formato *VQF* de *Yamaha* (mejor calidad y rendimiento en anchos de banda pequeños), *TAC* (mejor calidad y mayor compresión) o el novedoso *MPEG-2 AAC* (mejor calidad y compresión). Finalmente existe una última alternativa que va ganado lentamente adeptos y que pertenece a *Microsoft:* el formato *WMF*. Su única ventaja es que soporta formatos seguros de música que limitan la piratería de la misma; aunque queremos recordar que esta protección fue rota a las 24 horas escasas de su anuncio y que aún no existe un formato que lo evite.

CARACTERÍSTICAS TÉCNICAS

En los apartados anteriores afirmábamos que los archivos *MP3* están pensados para unas determinadas condiciones de uso y por lo tanto debemos saber qué factores son los que determinan este tipo de archivos, tanto en calidad como en el espacio y carga que requieren de la máquina.

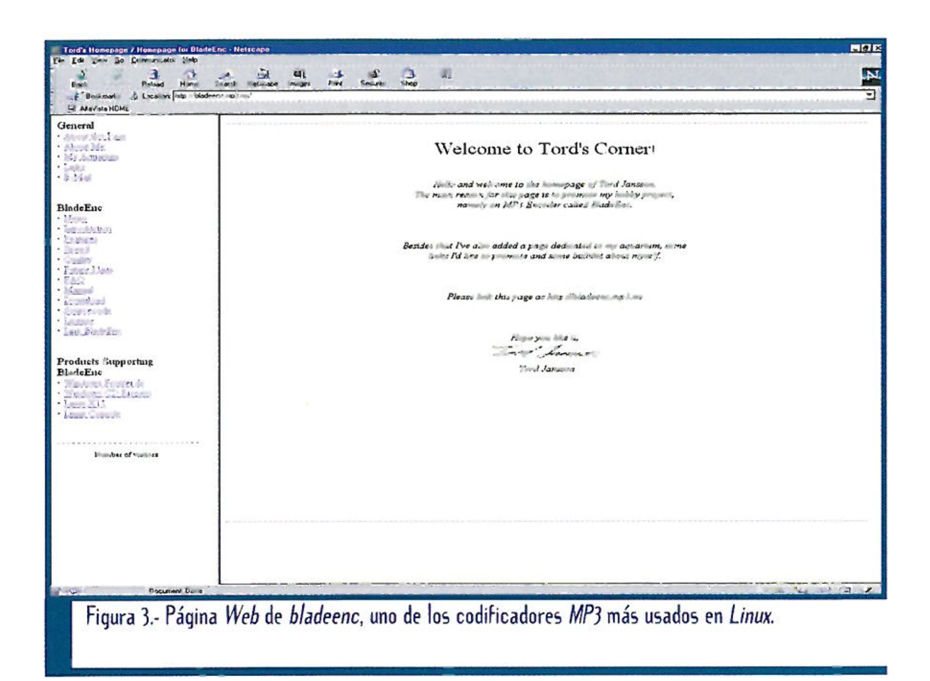


Figura 3.- Página *Web* de *bladeenc*, uno de los codificadores *MP3* más usados en *Linux*.

El primero de los factores que caracterizan un archivo *MP3* es el denominado ancho de banda, que determina el espacio que pude consumir en un segundo para representar lo más fielmente posible la información acústica. Este factor es medido en *Kbps,* y es la cantidad de Kilobits por segundo que puede consumir para representar dicha información (1 Kilobyte = 8 Kilobits). Este parámetro no puede tener cualquier valor sino que sólo puede tomar unos determinados valores que son: 8, 16, 24, 32, 40, 48, 56, 64, 80, 96, 112, 128, 144, 160 (para frecuencias de 16, 22 y 24 *Khz)* y 32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256, 320 (para frecuencias de 32, 44 y 48 *Khz).* En algunos compresores existe la limitación adicional de que los menores anchos de banda sólo son válidos para archivos monoaurales (un único canal de sonido).

El segundo parámetro que interviene en la creación de estos archivos es la frecuencia de muestreo. Su objetivo consiste en limitar el rango de frecuencias que van a ser codificadas con fidelidad. Esta limitación se basa en una propiedad del sonido que implica que para poder almacenar una onda sonora debemos digitalizarla al doble de su frecuencia. Teniendo en cuenta esta propiedad y sabiendo que la frecuencia máxima que puede percibir el oído humano está en torno



a los 20*Khz,* podemos entender que las frecuencias que mayor fidelidad ofrecen son 44 y 48 *Khz.*

Cabe destacar que en ámbitos profesionales no es raro hablar de frecuencias de muestreo de 96*Khz* o más, ya que aunque luego sean reducidas, la calidad es siempre mayor. En el mundo de los archivos *MP3* las frecuencias utilizadas son: 16, 22, 24, 32, 44 y 48*Khz.* Esta propiedad influye, además de en la calidad (por reproducir un rango mayor o menor de las frecuencias perceptibles por el oído), en el tamaño del archivo y, por lo tanto, en el aprovechamiento del ancho de banda ya que a mayor frecuencia, se necesitan más muestras para reproducir el sonido y el archivo aumenta de tamaño.

Dos importantes factores que determina la calidad final de un MP3 son el ancho de banda y la frecuencia de reproducción

Otro factor que determina el comportamiento de este tipo de archivos es lo que se conoce como espacialidad del sonido. Este término se refiere a la cantidad de canales que se utilizan para reproducir la música, que generalmente en el ámbito de los *MP3* suelen ser uno ó dos para el sonido. Al usarse dos canales multiplicamos por dos la cantidad de información necesaria para reproducir adecuadamente el mismo, ya que cada canal ha de ser almacenado de forma fiel.

Al utilizar un único canal, reducimos el espacio empleado, pero en el proceso perdemos la posibilidad de diferenciar el origen del sonido, y obtenemos un sonido plano, sin profundidad. Además la información contenida en los dos canales puede codificarse como información independiente o como una señal suma y una resta, obteniéndose en este caso mejores compresiones pero perdiendo en buena medida la espacialidad y profundidad del sonido estéreo.

El modelo marca unas reglas que todo compresor ha de cumplir, pero en ellas hay espacio suficiente para realizar optimizaciones

Como hemos mencionado antes, el estándar *MP3* define unas normas sobre cómo codificar la información, pero no obliga a utilizar el proceso de codificación perceptual predeterminado, sino que mientras se cumplan las normas definidas en el estándar, cualquier sistema de codificación perceptual es válido.

Esta situación ha llevado a la aparición de tres modelos diferentes de codificación perceptual, todos ellos reproducibles mediante un descodificador estándar. Estos modelos son:

- *ISO:* es el que se encuentra en la implementación de referencia del formato y es el que obtiene la peor calidad de los tres.
- *GPSYCHO:* es un modelo desarrollado de forma independiente y colocado bajo licencia *GPL* para que sea utilizado sin necesidad de pagar *royalties.* Su objetivo es modelar de forma más perfecta la percepción del oído humano y obtener de esta forma mejor calidad y mayor nivel de compresión; actualmente es utilizado por el codificador *Lame* de *Linux.*
- *PSYCHO:* está desarrollado por el Instituto *Fraunhofer,* uno de los integrantes del grupo *MPEG,* que ha desarrollado gran parte de la tecnología de

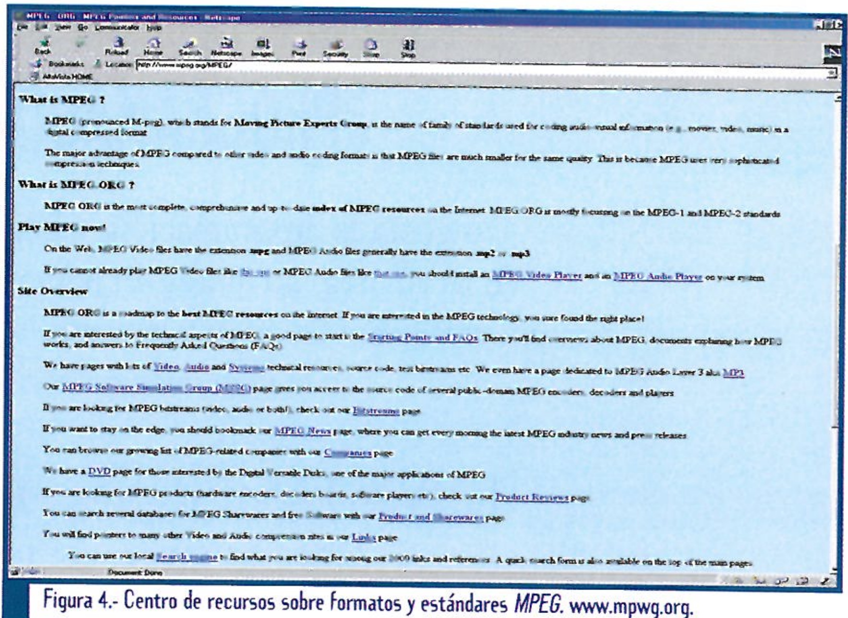


Figura 4.- Centro de recursos sobre formatos y estándares *MPEG.* www.mpwg.org.



audio utilizada en este formato. Su objetivo es el mismo que el del modelo *GPSYCHO*, pero a diferencia de éste, está bajo una patente y no es posible usarlo sin pagar *royalties* a dicho instituto.

Finalmente, un último factor que influye en el comportamiento y prestaciones de los archivos *MP3* es el uso del ancho de banda. En los inicios de este estándar la única forma de crear un archivo de este tipo era utilizando el mismo ancho de banda a lo largo del mismo, por lo que en ocasiones se desperdiciaba espacio al utilizarse más del que era requerido para obtener una adecuada representación del sonido, mientras que en otras ocasiones había fragmentos que no se reproducían con la calidad deseada.

Ante este problema apareció la codificación *VBR*, que va variando de forma automática el ancho de banda a lo largo del archivo, según las necesidades del mismo. De esta forma obtenemos archivos más pequeños que si utilizásemos un nivel superior de ancho banda, pero con mayor calidad que si estuviésemos usando el nivel inferior.

Ante todo debemos dejar claro que existen sistemas de compresión que ofrecen mejor calidad que el formato MP3

Estos factores son los que definen de forma más genérica el comportamiento de los archivos *MP3*, aunque hay compresores que permiten modificar otros parámetros más técnicos y que pueden ayudarnos mucho, pero sólo en situaciones extremadamente concretas, y no en general. Una vez que conocemos este comportamiento, podemos pasar a ver qué módulos básicos podemos implementar o pueden sernos de ayuda.

FORMAS DE USO DEL MP3

El uso de *MP3* por parte del programador requiere la implementación de unos algoritmos o subprogramas diferentes según vaya a ser el uso que vayamos a hacer de esta tecnología, independientemente de los requerimientos del sistema al que vayamos a ceñirnos. El hecho de utilizar *Linux* como base de estos desarrollos en *MP3* está dando muy buenos resultados, como podemos deducir del hecho de que muchos reproductores *hardware* (*Empeg*, por ejemplo) utilizan *Linux* como sistema operativo base, e incluso utilizan motores descodificadores o reproductores ya existentes en su funcionamiento.

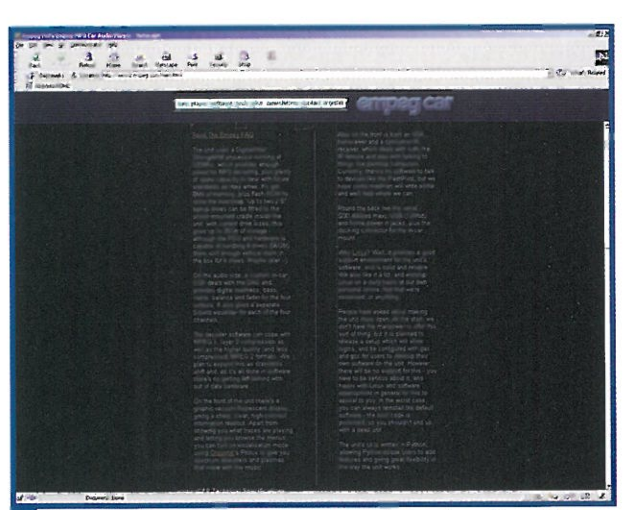


Figura 5.- Página *Web* de *empeg*, un reproductor de archivos *MP3* para coches basado en *Linux*.

En concreto, como desarrolladores, podemos necesitar incorporar alguna de estas funcionalidades:

- **Encoders o codificadores**
Son rutinas que, partiendo de la información original, la codifican utilizando el formato y las normas del *MP3*. La creación de estas funciones no es lo más frecuente por parte de un desarrollador, sino que es habitual utilizar la implementación de referencia o cualquier otra e incorporarla a nuestros programas.

- **Decoders o descodificadores**
Representan el paso contrario, permitendo restaurar la información en un formato digital sin compresión partiendo de un archivo *MP3*. Esta operación puede no ser en tiempo real y pueden incluirse diversos procesos para mejora de la calidad de la información generada, como puede ser la interpolación. En la actualidad no es una función especialmente trabajada, y -al igual que la anterior- es política habitual incorporar descodificadores ya creados y modificarlos, que no implementarlos desde cero.

- **Players o reproductores**
Estas rutinas parten de la información en *MP3* y la reproducen acústicamente en tiempo real. Desde el punto de vista del desarrollador son descodificadores que actúan en tiempo real, con las limitaciones que ello comporta, y suelen ser programas autónomos que incorporan numerosas prestaciones de cara a la reproducción en masa de archivos y/o al mantenimiento de colecciones musicales. Actualmente la mayoría de los

desarrollos en el mundo *Linux* se centran en este apartado.

Una vez que conocemos cuáles son las funcionalidades básicas que podemos implementar, podremos liberar nuestra imaginación y crear nuevas características que partan de éstas, como mezcladores de archivos *MP3,* aplicación de efectos, etc. En cualquier caso la imaginación es nuestro único límite.

Existen funcionalidades básicas de las que podemos derivar las demás, que son: reproducción, codificación y descodificación

En el caso que deseemos implementar nuestro propio codificador (para poder aprender más sobre las interioridades de este formato y ver cómo es posible optimizar su funcionamiento), son buenos puntos de partida los siguientes códigos fuente:

- **Implementación de referencia**

Es la creada por *ISO;* no es muy eficiente y no da la mejor calidad, pero está ampliamente comentada y se puede encontrar en *http://mp3tech.free.fr/programmers/sources/dist10.tgz* entre otros muchos sitios.

- **Lame**

Se trata de un codificador que contiene amplias modificaciones y mejoras respecto a la implementación de referencia. Su código fuente está disponible en *http://www.sulaco.org/mp3.*

- **Bladeenc**

Es un codificador también basado en la implementación *ISO,* pero que al igual que *Lame,* incorpora numerosas modificaciones. Este puede encontrarse en *http://bladeenc.mp3.no/.*

- **Gogo**

Éste está basado en *Lame,* pero contiene numerosas mejoras en cuanto a velocidad. Puede encontrarse en: *http://homepage1.nifty.com/herumi/gogo_e.html.*

Si nuestro objetivo es la creación de un descodificador o un reproductor y queremos ver o aprender de otros códigos, podemos partir de los siguientes puntos:

- **Implementación de referencia**

Creada por la *ISO,* es la menos eficiente, pero cuenta con una buena documentación interna.

- **MPEG123**

El más popular de todos los descodificadores *MP3* que hay en *Linux;* puede encontrarse en código fuente en *http//mpg.123.org.*

- **Open Cubic Player**

A diferencia de los anteriores, este programa no solamente contiene un motor de descodificación *MP3,* sino que tiene un motor de mezclas y una serie de rutinas de proceso del sonido muy interesantes. Este programa puede encontrarse en *http://www.cubic.org* pero hay que hacer notar que es bastante habitual que esta dirección se encuentre inactiva.

- **XMMS**

Es uno de los más populares reproductores *Linux* y es un buen modelo para ver cómo crear un reproductor ya que es muy completo y además tiene un módulo de ecualización. Este programa puede encontrarse en *http://www.xmms.org.*

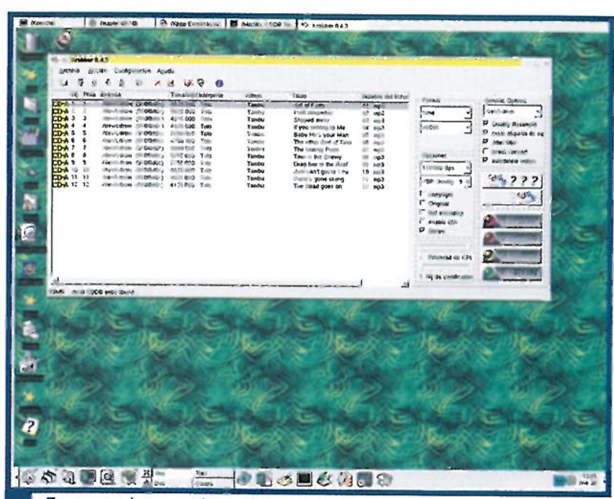


Figura 6.- Aspecto de un *front-end,* programa que nos ayuda a trabajar con los archivos *MP3.*

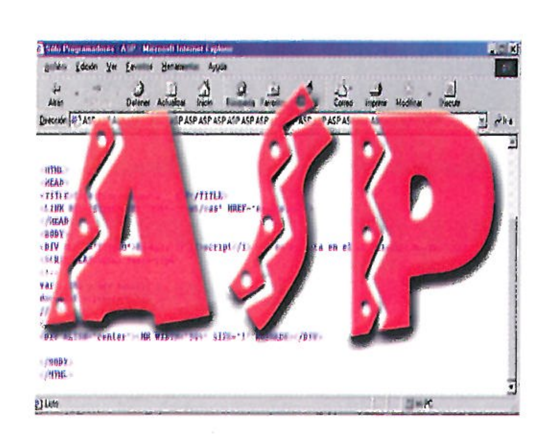
Adicionalmente es posible encontrar librerías que nos ayuden en la creación de programas que utilicen este formato de una u otra forma en nuestra máquina *Linux,* pero esto lo veremos en el próximo artículo.

CONCLUSIÓN

En este artículo hemos querido hablar de la parte más teórica de tecnología *MP3,* ya que resulta necesario para poder realizar un uso correcto de esta tecnología en nuestros propios programas. Así mismo hemos pretendido dar unos puntos de partida para aquéllos que quieran adelantarse para encontrar recursos e información les permita investigar por su cuenta.

En la próxima entrega de esta serie nos centraremos más en los aspectos prácticos y veremos cómo incluir alguna de estas funcionalidades en el interior de nuestras aplicaciones.





La tecnología ASP (y VI): ASP, XML y XSL

Adolfo Aladro García. *Analista programador*

El lenguaje *XSL* es el complemento perfecto para las fuentes de datos en formato *XML*, puesto que permite presentar los documentos *XML* de una forma tan sofisticada como se desee. Esto es posible gracias a la capacidad de *XSL* para procesar y tratar los datos *XML*. Hoy en día las páginas *ASP* presentan uno de los entornos más adecuados donde empezar a implantar todas estas tecnologías.

COMPONENTES ACTIVEX DE SERVIDOR

En el capítulo anterior se analizó la integración de las fuentes de datos *XML* dentro de las páginas *ASP*. El siguiente paso para sacar el mayor provecho posible a la tecnología *XML* consiste en la utilización de las hojas *XSL* para tratar y procesar los datos. Si bien es cierto que el propio *DOM* de *XML* permite manipular los datos mediante *JScript* con bastante flexibilidad, el lenguaje *XSL* es el medio natural para realizar estas tareas.

Los documentos *XML* solamente contienen datos y por lo tanto no hay nada en ellos que diga algo acerca de cómo se deben presentar. No en vano esta es la razón de ser de la tecnología *XML*: separar la presentación de los contenidos. El lenguaje *XSL* surge con la intención de proporcionar una vía a través de la cual los datos *XML* se visualicen dentro de las páginas *Web*.

LAS HOJAS XSL

El lenguaje *XSL* toma los datos procedentes de un documento *XML* y da como resultado una transformación de los mismos normalmente en forma de código *HTML* listo para ser insertado dentro de una página *Web*. Durante el proceso de tratamiento es posible ordenar la información, generar texto adicional o realizar todo tipo de cálculos. Esto no significa que se modifique la fuente de datos original, es decir, el documento *XML* de partida.

El lenguaje XSL sirve para presentar los datos

Se define una hoja *XSL* como un documento que contiene el código



necesario para analizar la estructura de una fuente de datos *XML* y tratarla. Podría decirse que una hoja *XSL* es algo así como una plantilla que contiene instrucciones precisas para generar el código *HTML* necesario a partir de un documento *XML*. La filosofía de las páginas *ASP* y la de las hojas *XSL* es bastante parecida.

El código que sigue es un sencillo ejemplo de un documento *XML*:

```
<?xml version="1.0"?>
<discos>
  <disco>
    <artista>Madonna</artista>
    <titulo>Ray Of Light</titulo>
  </disco>
</discos>
```

Esta fuente de datos no puede aparecer tal cual dentro de una página *Web* ya que el navegador no sabría qué hacer con ella.

Las hojas XSL leen y tratan los datos XML

La siguiente hoja *XSL* podría servir para visualizar el documento *XML* anterior:

```
<?xml version="1.0" ?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/
    TR/WD-xsl">
  <xsl:template match="/">
    <FONT SIZE="+3"><B>
      <xsl:value-of select="
      /discos/disco/artista"/><BR/>
      <xsl:value-of select="
      /discos/disco/titulo"/>
    </B></FONT>
  </xsl:template>
</xsl:stylesheet>
```

Internet Explorer 5.0 es por el momento el único navegador que soporta los lenguajes *XML* y *XSL* así que antes de pasar a ver cómo es posible incorporar las páginas *XSL* a las páginas *ASP* es posible visualizar el resultado de aplicar la hoja *XSL* al documento *XML* simplemente añadiendo la siguiente línea inmediatamente después de la primera:

```
<?xml-stylesheet
    type="text/xsl"
    href="discos.xsl" ?>
```

Figura 1.- Resultado de visualizar el archivo **discos.xsl** directamente con *Internet Explorer* 5.0.

COMUNICACIÓN POR MENSAJES

Después de haber echado un primer vistazo al lenguaje *XSL* resulta curioso comprobar el parecido en cuanto a la sintaxis entre un documento *XSL* y otro cualquiera de *XML*. En realidad el lenguaje *XSL* constituye en sí mismo un subconjunto del lenguaje *XML*. Es decir, todas aquellas reglas que han de cumplirse para poder considerar que un documento *XML* está bien formado también se aplican a las hojas *XSL*. Brevemente recordamos esas reglas:

- Todas las etiquetas deben cerrarse.
- El anidamiento de las etiquetas debe ser el correcto.
- Se distingue entre mayúsculas y minúsculas.
- Los atributos deben ir siempre entre comillas.
- Solamente puede haber una etiqueta raíz.
- Entidades de caracteres.
- Bloques de código.

Una hoja XSL es en realidad un documento XML

En las hojas *XSL* se intercalan instrucciones del lenguaje *XSL* con etiquetas *HTML*. El lenguaje *HTML* es también un subconjunto determinado de *XML* pero dada la flexibilidad de los navegadores a la hora de interpretarlo, con frecuencia las páginas *HTML* no son documentos *XML* bien formados. Por esta razón todo código *HTML* que aparezca en las hojas *XSL* debe cumplir las siete normas anteriores. Esta es la razón por la que la etiqueta *
* aparece como *
*. De no ser así el *parser* de *XML* analizaría la hoja *XSL* y daría un error. Para probar esto podemos utilizar de nuevo *Internet Explorer 5.0*.

La Figura 3 muestra la respuesta del navegador al tratar de mostrar directamente el archivo **discos.xsl** en el que hemos puesto la etiqueta *
* como *
*. Aunque esto es válido dentro del lenguaje *HTML*, no lo es para el *XML*.

Todo lo anterior suele ser normalmente motivo de confusión y al final el lector termina no teniendo nada claro la diferencia entre *XML*, *XSL* y *HTML*. Por ello vamos a repasar concisamente la definición de estos tres lenguajes, matizando las relaciones que hay entre ellos.

- *XML*: El lenguaje *XML* sirve para describir datos. Los documentos *XML* pueden considerarse como pequeñas bases de datos.

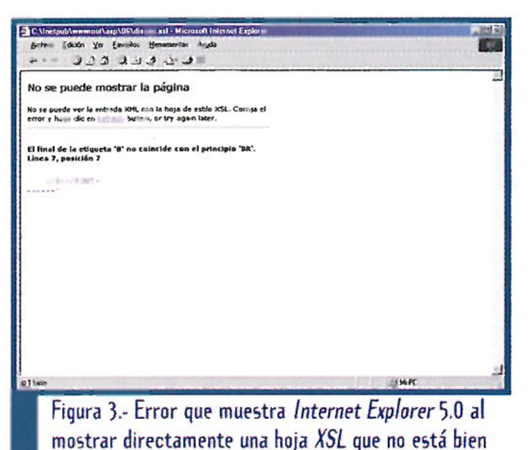


Figura 3.- Error que muestra *Internet Explorer* 5.0 al mostrar directamente una hoja *XSL* que no está bien formada.

- *HTML*: El lenguaje *HTML* sirve para presentar la información. No obstante podría considerarse como un subconjunto específico de *XML* ya que su sintaxis es prácticamente idéntica a la propuesta por el lenguaje *XML*. El problema que existe al hacer esta valoración reside en el hecho de que las páginas *HTML* no suelen respetar las normas que garantizan que un documento *XML* esté bien formado.

- *XSL*: El lenguaje *XSL* sirve para presentar la información procedente de una fuente de datos *XML*. Al igual que ocurre con el lenguaje *HTML*, *XSL* es un subconjunto específico de *XML*. Para que una hoja *XSL* pueda interpretarse correctamente ha de ser un documento *XML* bien formado. El lenguaje *XSL* ya contempla esta restricción pero dentro de las hojas *XSL* suelen aparecer con frecuencia bloques de código *HTML*, y como ya hemos apuntado, el lenguaje *HTML* presenta muchos casos en los que no se cumplen las siete normas. Por ello, cuando se escribe una hoja *XSL* hay que "forzar" el código *HTML* para que se cumplan (de ahí que se escriba *
* en vez de *
*; o por ejemplo *<IMG SRC="p.gif"/>* en vez de *<IMG SRC=p.gif>*, que sería perfectamente válido en una página *HTML* normal y corriente).

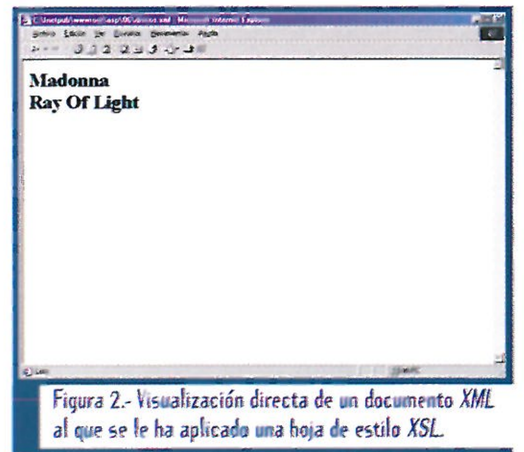


Figura 2.- Visualización directa de un documento *XML* al que se le ha aplicado una hoja de estilo *XSL*.

XSL Y LAS PÁGINAS ASP

Las hojas *XSL* pueden aplicarse en el lado del servidor, mediante las páginas *ASP*. Esta solución es obviamente mucho mejor ya que en el lado del cliente el único navegador que por el momento soporta estas tecnologías es *Internet Explorer* 5.0. Veamos cómo se hace esto a través de un pequeño ejemplo.

Los bloques de código HTML de una hoja XSL tienen que estar bien formados

En primer lugar debemos localizar los archivos **buzon.xml** y **buzon.xsl** que contienen respectivamente los datos y la hoja *XSL* que se va a utilizar para presentarlos:

```
var fichero_xml =
    Server.MapPath("buzon.xml");
var fichero_xsl =
    Server.MapPath("buzon.xsl");
```

Las variables **fichero_xml** y **fichero_xsl** tienen a partir de este momento una referencia a esos archivos. Cuando hablamos de "localizar" nos estamos refiriendo a la correspondencia que ha de hacer el servidor entre direcciones virtuales y direcciones reales en el disco duro del ordenador.

El siguiente paso consiste en cargar esos archivos para que puedan ser procesados por el servidor adecuadamente:

```
var datos_xml =
    Server.CreateObject("Microsof
    t.XMLDOM");
datos_xml.async = false;
datos_xml.load(fichero_xml);

var datos_xsl =
    Server.CreateObject("Microsof
    t.XMLDOM");
datos_xsl.async = false;
datos_xsl.load(fichero_xsl);
```

Las variables **datos_xml** y **datos_xsl** contienen ahora referencias a la fuente de los datos *XML* y a la hoja *XSL* respectivamente. Este proceso muestra con toda claridad que las hojas *XSL* son a su vez documentos *XML* – especiales, pero documentos *XML* al fin y al cabo - y como tales son tratados por el código de las páginas *ASP*. El mismo núcleo encargado de cargar y procesar los documentos *XML* hace lo propio con las hojas *XSL*.

El último de los pasos consiste en utilizar el *DOM* de *XML* para significar que queremos mostrar los datos *XML* mediante esa hoja *XSL*.




```
<%=
     datos_xml.transformNode(datos
     _xsl) %>
```

El método *transformNode* de un documento *XML* recibe como parámetro una referencia a una hoja *XSL* y devuelve el código resultante de aplicar la hoja *XSL* al documento *XML*. Este código resultante normalmente suele ser *HTML*.

El método transformNode sirve para aplicar una hoja XSL a un documento XML

Como hemos podido apreciar la utilización de hojas *XSL* con fuentes de datos *XML* y dentro de las páginas *ASP* es muy sencilla. Sin embargo hasta ahora no hemos hecho más que aproximarnos a la potencia que tienen estas tecnologías. Las posibilidades son muchas y por lo tanto sacar el máximo partido de estos lenguajes requiere un estudio pormenorizado del lenguaje *XSL* y del *DOM* de *XML*.

En el capítulo anterior de esta serie se vió el *DOM* de *XML*. A continuación vamos a resumir algunas de las características más importantes del lenguaje *XSL*. No obstante recomendamos una vez más a todos los lectores interesados que visiten *msdn.microsoft.com /workshop*.

Figura 5.- Visualización directa del documento *XML* buzon.xml.

EL LENGUAJE XSL

Si nos fijamos en el código fuente de los archivos **buzon.xsl** o **discos.xsl**, utilizados en los ejemplos anteriores, veremos que se trata de una amalgama compuesta de código *HTML* y etiquetas propias del lenguaje *XSL*. Estas etiquetas constituyen un lenguaje mediante el cual podemos recorrer los datos de las fuentes representadas por los documentos **buzon.xml** o **discos.xml**.

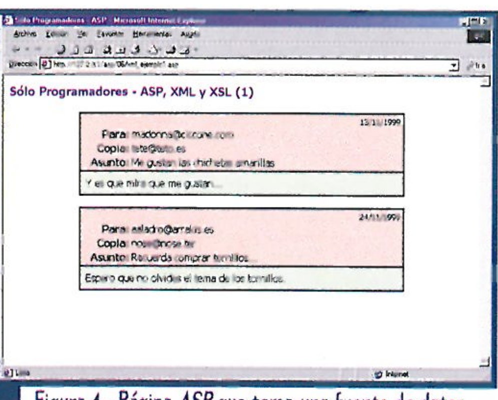


Figura 4.- Página *ASP* que toma una fuente de datos *XML* y una hoja *XSL* para presentar la información.

Antes de pasar a ver algunas de las etiquetas más importantes del lenguaje *XSL* es preciso analizar cómo éste es capaz de reconocer e interactuar con las estructuras presentes en una fuente de datos *XML*.

LOS PATRONES XSL

Se denomina patrón *XSL* a una cadena de texto que representa una referencia a un nodo del documento *XML*. Esta referencia puede hacerse teniendo criterios tales como el nombre de un nodo, su valor, el tipo, etc.

```
<xsl:value-of select="/dis-
cos/disco/artista"/>
```

La etiqueta anterior que aparecía en la hoja **discos.xsl** contiene un patrón:

```
/discos/disco/artista
```

La estructura de los documentos *XML* se asemeja a la de directorios y subdirectorios de un disco duro. Un patrón sería por lo tanto una ruta dentro de ese disco duro hipotético. El carácter / sirve separar los elementos entre sí. De esta forma el patrón anterior se refiere claramente al nodo artista dentro del documento *XML*.

La sintaxis de los patrones *XSL* puede complicarse considerablemente. Veamos algunos ejemplos:

- **/discos/*/artista**

Este patrón sería válido para identificar a todos los nodos del documento *XML* etiquetados como **artista**. El carácter * indica que éstos no tienen porqué ser necesariamente hijos de nodos etiquetados como **disco**, tal y como expresaba el patrón anterior (**/discos/disco/artista**).

Nota: evidentemente esto no tiene sentido alguno a tenor del documento XML de partida. Éste y los ejemplos que veremos a continuación tienen solamente una función didáctica. Algunos tendrán sentido y otros no. Se trata solamente de mostrar el uso de los patrones

- **/discos/disco[titulo]/artista**

Este patrón representa una bifurcación. Sería válido para todos aquellos nodos etiquetados como **artista**, siempre y cuando su nodo padre, **discos**, tuviera un nodo hijo etiquetado como **titulo**.

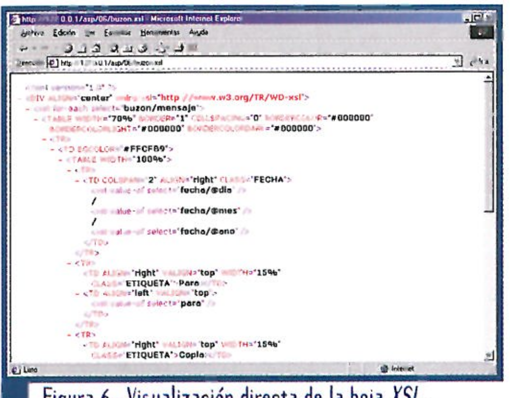


Figura 6.- Visualización directa de la hoja *XSL*, **buzon.xsl**, que utiliza para presentar los datos del documento **buzon.xml**

- **/discos/disco[titulo='Ray Of Light']/artista**

Las bifurcaciones también puede realizarse añadiendo comparaciones. Este patrón es similar al anterior pero con la condición adicional que exige que el nodo etiquetado como **titulo** almacene el valor **Ray Of Light**.

- **/buzon/mensaje[@id ="2134"]/asunto**

El carácter "@" se utiliza para hacer referencia a un atributo de una etiqueta *XML*. Este patrón, aplicado esta vez a la fuente de datos *XML* **buzon.xml**, sería válido para todos aquellos nodos etiquetados como **asunto** cuyo nodo padre, **mensaje**, tuviera un atributo etiquetado como **id** con valor igual a **2134**.

Los patrones son un método de acceso a los nodos XML

La sintaxis de patrones que proporciona el lenguaje *XSL* es muy potente y permite realizar todo tipo de búsquedas condicionales. Aquí hemos señalado tan sólo algunas de las posibilidades más sencillas. En la dirección: *msdn.microsoft.com /xml/xsl/reference/XSLPatternSyntax.asp* podemos encontrar una referencia completa al respecto.

LAS ETIQUETAS XSL

Las etiquetas *XSL* son al lenguaje *XSL* lo que las sentencias *for*, *if-then-else*, *while*, etc., son al lenguaje *C*. Con ellas es posible construir bloques de código que trabajen directamente sobre los datos *XML*. Los patrones representan la forma en la que las etiquetas pueden identificar los datos que van a ser tratados.

- **xsl:stylesheet**

Esta etiqueta se utiliza para significar que se trata de una hoja *XSL*.

- **xsl:template**

Antes de pasar a realizar algún tipo de operación con los datos *XML* es necesario indicar mediante un patrón a qué datos nos estamos refiriendo dentro del árbol. Esta etiqueta se utiliza para significar que se va a aplicar a los datos indicados por el patrón el estilo definido por el código *XSL* contenido entre las etiquetas *xsl:template* de apertura y cierre. Su sintáxis es la que sigue:

```
<xsl:template language="lenguaje"
    match="contexto" >
```

Por el momento el atributo *language* podemos omitirlo. El atributo *match* contiene el patrón que indica los datos que van a ser tratados.

XSL cuenta con sentencias para hacer bucles y bloques de código condicionales

- **xsl:for-each**

En el documento **buzon.xsl** encontramos un ejemplo ilustrativo de la utilización de esta etiqueta.

```
<xsl:for-each select="buzon/mensa-
    je">
...
</xsl:for-each>
```

El bloque anterior sirve para recorrer todos los nodos etiquetados como **mensaje** que son hijos del nodo etiquetado como **buzon.** La sintaxis de esta etiqueta es la siguiente:

```
<xsl:for-each order-by="criterio-
    ordenación" select="patrón">
```

Las etiquetas XSL son sentencias de un lenguaje de programación

Una de las cosas más interesantes de esta etiqueta es el atributo *order-by*, que podrá utilizarse para ordenar los datos. Contiene una cadena de texto que constituye una lista de criterios de ordenación separados por el carácter ";". Los criterios se aplican en el orden en el que aparecen. Al tratar de ordenar dos elementos si el primer criterio no proporciona ninguna información, automáticamente se intenta aplicar el segundo criterio y así sucesivamente. El primer caracter distinto de blanco que aparece en cada crite-

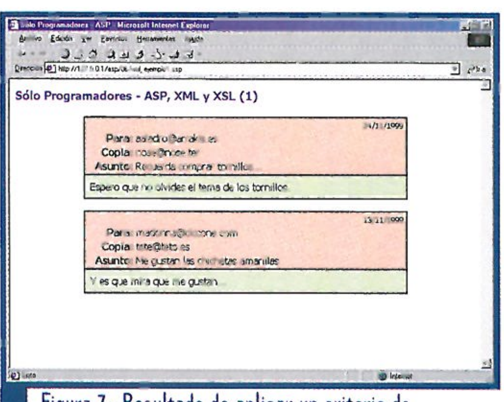


Figura 7.- Resultado de aplicar un criterio de ordenación en la hoja *XSL* **buzon.xsl** encargada de presentar los datos.



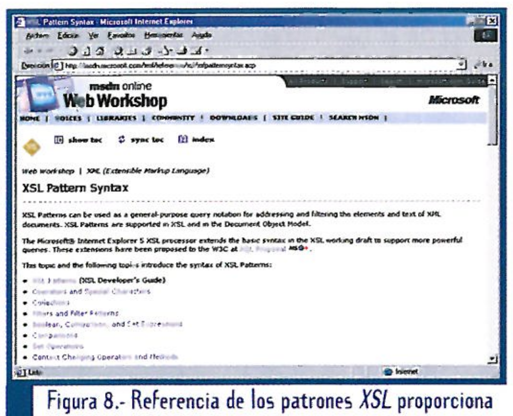


Figura 8.- Referencia de los patrones *XSL* proporciona en el sitio *Web* de *Microsoft*.

rio determina si se sigue un orden ascendente (+) o descendente (-). El criterio de ordenación se expresa como un patrón *XSL* relativo al patrón descrito en el atributo *select*.

Así por ejemplo podríamos modificar la hoja *XSL* **buzon.xsl** aplicando el siguiente criterio:

```
<xsl:for-each select="buzon/mensa-
     je" order-by="-@id">
...
</xsl:for-each>
```

Otro ejemplo de ordenación podría ser el que sigue:

```
<xsl:for-each select="buzon/mensa-
     je" order-by="-fecha/@dia">
...
</xsl:for-each>
```

El atributo order-by representa un método potente de ordenación

Es muy importante comprender que el patrón que aparece en el atributo *order-by* se analiza en función del contexto marcado por el patrón del atributo *select*. Recurriendo de nuevo al símil de los directorios y subdirectorios de un disco duro, sería como tener en cuenta que el atributo *select* marca un directorio absoluto y todas las referencias que aparecen en el atributo *order-by* son relativas a él.

- **xsl:value-of**

Esta etiqueta sirve para extraer el valor de un determinado. Este valor se integra en el resultado final que devuelve la hoja *XSL* cuando se aplica al documento *XML*.

```
<xsl:value-of
select="fecha/@dia"/>
```

El ejemplo anterior, que se encuentra dentro del bucle *xsl:for-each* principal del documento **buzon.xsl**, indica que queremos evaluar el valor del atributo *dia* de la fecha del mensaje que esté siendo procesado en ese momento.

Obsérvese que esta etiqueta es única, es decir, no tiene etiqueta de apertura y etiqueta de cierre, por lo que, teniendo en cuenta las normas, es necesario terminarla con el carácter / antes de que se cierre.

- **xsl:if**

Todos los lenguajes de programación presentan sentencias condicionales. Esta etiqueta representa la sentencia condicional por excelencia del lenguaje *XSL*.

En la hoja *XSL* podríamos aplicar la siguiente condición inmediatamente después de empezar el bucle principal:

```
<xsl:if
    test="@id[.='2134']">···</xsl
    :if>
```

De esta manera estaríamos indicando que sólo nos interesa el nodo etiquetado como **mensaje** cuyo atributo **id** tiene un valor igual a **2134**.

Otro ejemplo podría ser el siguiente:

```
<xsl:if test="fecha/@dia[. $le$
    30]">··· </xsl:if>
```

La sentencia anterior haría que sólo se mostrasen los mensajes en los que el día de la fecha fuera menor que **30**.

Las condiciones de la sentencia *xsl:if* se expresan mediante patrones, y por lo tanto pueden ser tan complejas como sea necesario.

- **xsl:choose, xsl:otherwise y xsl:when**

Estas etiquetas se utilizan también para construir bloques de código condicionales. Su sintaxis es similar a la que exhibe la etiqueta *xsl:if* y representan el equivalente a la sentencia *switch* de lenguajes como *Java* o *JavaScript*. Así por ejemplo podríamos hacer:

```
<xsl:choose>
  <xsl:when test="@id[.='2134']">
  ...
  </xsl:when>
<xsl:otherwise>
  ...
</xsl:otherwise>
</xsl:choose>
```

CONCLUSIÓN

Con estos dos últimos capítulos de la serie sobre la tecnología *ASP* hemos introducido los lenguajes *XML* y *XSL*. Nunca se puede asegurar nada acerca de las tecnologías que finalmente se impondrán en un medio tan voluble como *Internet* pero lo que podemos afirmar con contundencia es que *XML* presenta unas perspectivas de futuro nada desdeñables. Aquí nos hemos encontrado simplemente con el principio de la hebra, pero las posibilidades son tantas y tan variadas que hay cuerda para rato.

¿TE FALTA ALGÚN NÚMERO DE...?

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SÓLO PROGRAMADORES

ATENCIÓN

Los números 49 al 51, ambos inclusive,

se encuentran AGOTADOS

























BOLETÍN DE PEDIDOS

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L. (Revista Sólo Programadores). C/ San Sotero, 5. 1ª Planta. 28037 Madrid
Tlf: 91 304 87 64. Fax: 91 327 13 03

Deseo que me manden los número/s...............................
....
....
NOMBRE Y APELLIDOS:...
....
EDAD:........... PROFESIÓN: ....TFNO: ....
DOMICILIO: ....
CIUDAD: ....
PROVINCIA: ....C.P: ....

❑ **Precio por unidad: 975 pesetas** + 700 ptas. de gastos de envío.

FORMAS DE PAGO:

❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❑ Talón bancario a nombre de REVISTAS PROFESIONALES. S.L
❑ Domiciliación bancaria
❑ Contra reembolso

DATOS DE DOMICILIACIÓN:

Banco...

FIRMA

Domicilio..
Nº de Cuenta..............|..............|..............|.............................
Titular...
Fecha..





Visual C++ y MFC (II)

Constantino Sánchez Ballesteros.
Técnico Superior Dearrollo Aplicaciones Informáticas.

Continuamos el desarrollo del programa iniciado el pasado mes para adentrarnos en el control de componentes ActiveX dentro de una aplicación. Utilizaremos la caja de texto, un botón de comando y varias fuentes de letra.

PROGRAMANDO, PROGRAMANDO...

Vamos a comenzar nuestras primeras líneas de código haciendo que nuestra ventana aparezca siempre encima de las demás aunque pierda el foco (comúnmente se conoce como *TOPMOST).* Para realizar este proceso, lo primero que tenemos que saber es en qué lugar efectuarlo.

La clase *CMainFrame* derivada de *CFrameWnd* tiene la función *PreCreateWindow* que se utiliza para determinar las características de una ventana antes de que se visualice. Centrándonos en nuestro programa, si abrimos el archivo **MainFrm.cpp** encontraremos esta función. También podemos llegar a ella utilizando *ClassView*, tal y como se muestra en la Figura 1.

La construcción de la función es la siguiente:

```
BOOL
CMainFrame::PreCreateWindow(
CREATESTRUCT& cs)
{
    if(
!CFrameWnd::PreCreateWindow
(cs) )
        return FALSE;
    // TODO: Modify the
Window class or styles here
    // by modifying the CREA-
TESTRUCT cs
    return TRUE;
}
```

La variable **cs** de *CREATESTRUCT* permite definir las propiedades de la ventana. La propiedad *dwExStyle* de la variable **cs** ofrece la posibilidad de establecer características especiales para una ventana. En nuestro caso, debemos asignar a esta propiedad el valor

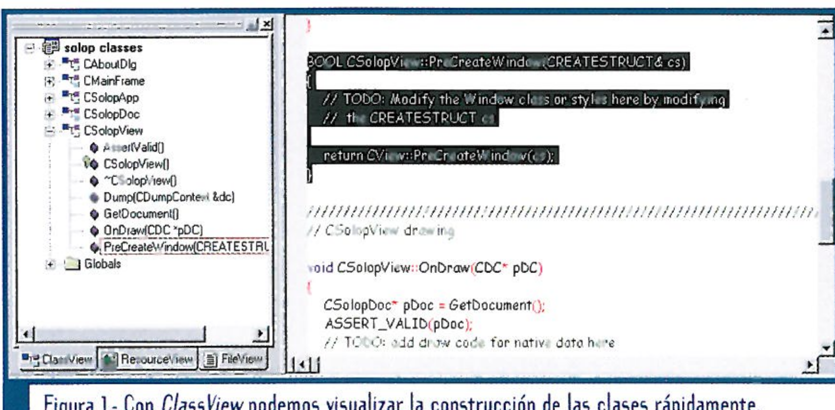


Figura 1.- Con *ClassView* podemos visualizar la construcción de las clases rápidamente.



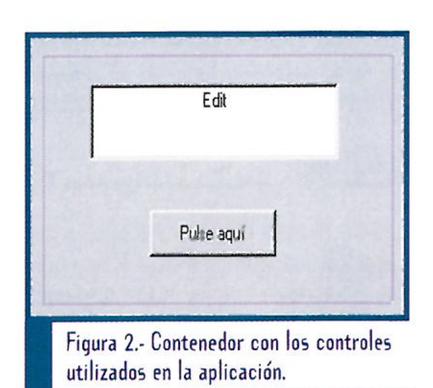


Figura 2.- Contenedor con los controles utilizados en la aplicación.

WS_EX_TOPMOST, que permitirá establecer la ventana en el modo que buscamos.

Una vez realizado este proceso, el extracto de código anterior quedaría como sigue:

```
BOOL
CMainFrame::PreCreateWindow(
CREATESTRUCT& cs)
{
    if(
!CFrameWnd::PreCreateWindow
(cs) )
        return FALSE;
    // TODO: Modify the
Window class or styles here by
    // modifying the CREATES-
TRUCT cs

    cs.dwExStyle =
WS_EX_TOPMOST;

    return TRUE;
}
```

A continuación podemos ver las características especiales que podemos aplicarle a una ventana:

- *WS_EX_ACCEPTFILES:* especifica que una ventana acepta la característica *drag-and-drop* en archivos (arrastrar y soltar).
- *WS_EX_CLIENTEDGE:* la ventana tendrá un aspecto *3D* con un borde.
- *WS_EX_CONTEXTHELP:* incluye una interrogación en la barra de título de la ventana.
- *WS_EX_CONTROLPARENT:* permite al usuario navegar por las ventanas hijas de la ventana principal mediante la tecla **Tab.**
- *WS_EX_DLGMODALFRAME:* designa una ventana con un borde doble.
- *WS_EX_LEFT:* establece en la ventana propiedades de alineamiento a la izquierda.
- *WS_EX_LEFTSCROLLBAR:* coloca una barra de *scroll* vertical a la izquierda del área cliente.
- *WS_EX_LTRREADING:* visualiza el texto de la ventana utilizando el orden de lectura de izquierda a derecha. Es una propiedad establecida por defecto.
- *WS_EX_MDICHILD:* crea una ventana hija de la principal.
- *WS_EX_OVERLAPPEDWINDOW:* combina los estilos *WS_EX_CLIENTEDGE y WS_EX_WINDOWEDGE.*
- *WS_EX_PALETTEWINDOW:* combina los estilos *WS_EX_WINDOWEDGE y WS_EX_TOPMOST.*
- *WS_EX_RIGHT:* establece en la ventana las propiedades de alineamiento a la derecha. Esto depende de la clase de la ventana.
- *WS_EX_RIGHTSCROLLBAR:* coloca una barra de *scroll* vertical (si está presente) a la derecha del área cliente. Es un valor por defecto.
- *WS_EX_RTLREADING:* visualiza el texto de la ventana de derecha a izquierda.
- *WS_EX_STATICEDGE:* crea una ventana con un borde tridimensional.
- *WS_EX_TOOLWINDOW:* crea una ventana de herramientas, comúnmente utilizada como barra de herramientas flotante.
- *WS_EX_TOPMOST:* una ventana con este estilo se colocará siempre por encima de las demás, aunque pierda el foco.
- *WS_EX_TRANSPARENT:* hará que la ventana sea transparente y no se visualizará.

FUNCIONAMIENTO INTERNO DE WINDOWS

Una aplicación *Windows* utiliza, al menos, una ventana para que el usuario pueda comunicarse con ella. Una interfaz de este tipo generalmente se diseña para permitir al usuario realizar tanto operaciones de *E/S* como para poner en marcha las distintas operaciones para las que fue programada la aplicación.

PreCreateWindow se utiliza para determinar las características de una ventana antes de que se visualice

La aplicación es responsable de crear la ventana y comparte la manipulación de la misma con *Windows.* Esto significa que *Windows* es responsable de manipular el tamaño, la posición y los controles de la ventana, mientras que la aplicación manipula el área de trabajo del usuario.

COMUNICACIÓN POR MENSAJES

Un mensaje es una notificación que *Windows* envía a una aplicación en ejecución para indicarla que ha sucedido algo; por ejemplo, una acción del usuario, como un clic sobre un botón, mover el ratón, pulsar una tecla, elegir una orden

Figura 3.- Cambio de clase para la vista del programa.

de un menú, modificar el tamaño de una ventana, etc; una acción de la propia aplicación, como mandarse un mensaje a sí misma forzando su minimización, o una acción del propio *Windows,* como enviar un mensaje a una aplicación para que redibuje su ventana. Lógicamente la aplicación responderá al suceso ocurrido con una acción específica determinada por la ejecución del código de una función. Esto significa que mandar un mensaje equivale a ejecutar una función dentro de la aplicación.

El sistema de mensajes de *Windows* hace posible que distintas tareas compartan el procesador, característica que define un sistema multitarea. *Windows* forma una cola con los mensajes que se producen y los distribuye a las aplicaciones correspondientes. Cada aplicación que recibe un mensaje, simplemente tiene que ejecutar el proceso asociado.

INDEPENDENCIA DEL HARDWARE

Para que *Windows* mantenga el control de la multitarea tiene que estar entre la aplicación y el *hardware.* Sólo así puede interceptar cualquier entrada del usuario y enviar el mensaje correspondiente a la aplicación apropiada. Por este motivo, para que una aplicación tenga contacto con el *hardware* habrá de ser a través de *Windows.*

Por ejemplo, una aplicación *Windows* no escribe directamente sobre la impresora sino que utiliza las rutinas apropiadas del *Kit de Desarrollo de Software (SDK)* para hacerlo. Esto implica que los fabricantes de *hardware* que quieran que sus productos funcionen con *Windows* deben atenerse a los estándares que se definen en el *Microsoft Windows OEM Adaptation Kit* que proporciona *Microsoft* y que definen las capacidades mínimas que debe tener el hardware para asegurar un funcionamiento correcto de las rutinas de *SDK.*

WS_EX_ACCEPTFILES especifica que una ventana acepta la característica drag-and-drop en archivos

En lo referente a la salida, cada rutina de *SDK* es capaz de dividirse en los conjuntos mínimos de operaciones que un dispositivo requiere. Con respecto a la entrada, *Windows* tiene predefinida toda entrada de ratón y de teclado (teclado virtual *Windows)* que una aplicación puede recibir. Por lo tanto, si un fabricante realiza un teclado que contenga teclas diferentes a las del teclado virtual de *Windows* o diseña un ratón con cuatro botones por ejemplo, debe también facilitar el *software (drivers)* de conversión necesario.

FUNCIONES

Las funciones *Windows* son el corazón de las aplicaciones *Windows.* Hay cientos de ellas dispuestas para ser llamadas desde *C/C++.* Todas las funciones están declaradas en un fichero de cabecera denominado windows.h incluido en el *SDK* y por lo tanto en *Visual C++.*

VENTANAS Y PROCEDIMIENTOS DE VENTANA

Como hemos venido observando, las ventanas son objetos con unas propiedades y un código asociado. Esto implica que la programación en *Windows* sea una programación orientada a objetos.

La ventana de una aplicación generalmente contiene un título que se corresponde con el título de la aplicación, un menú, bordes de tamaño y barras de desplazamiento cuando sean necesarias. A su vez, esta ventana puede desplegar otras ventanas adicionales denominadas cuadros o cajas de diálogo, que contienen controles como botones, cajas de texto, etiquetas, etc., denominados también ventanas hijas.

Cada clase de ventana creada por una aplicación tiene asociado un procedimiento de ventana. Este procedimiento es una función incluida en la propia aplicación o en una biblioteca dinámica que recibe y procesa mensajes. Una vez procesado el mensaje, el control es devuelto a *Windows.* Dicho de otra forma, la ventana recibe la entrada del usuario en forma de mensaje. Por otra parte, es evidente que ventanas diferentes pueden reaccionar de forma distinta ante mensajes del mismo tipo, y que para



manejar cada tipo de mensaje es necesaria una rutina. Quiere esto decir que cada ventana tendrá su propio conjunto de rutinas. Estas rutinas agrupadas forman el procedimiento de ventana.

TRABAJO CON CONTROLES

Es el momento de empezar a insertar y utilizar algunos de los controles *ActiveX* que se incluyen con *Visual C++*. Por supuesto, es posible utilizar cualquier control que tengamos registrado en *Windows* y tengamos permiso para su utilización en nuestros programas.

Por defecto, tenemos a nuestra disposición los controles comunes visualizados en programas para *Windows* (botones, cuadros combinados, cuadros de lista, imágenes, etc).

Para iniciar el proyecto utilizaremos el asistente. Para ello, iremos al menú **File** y seleccionaremos **New**. En nuestro caso en particular debemos seleccionar **MFC AppWizard (EXE)** desde la pestaña *Projects*. Una vez dado el nombre deseado al proyecto y seleccionar el directorio de trabajo correspondiente, aparecerá una primera pantalla del asistente que nos pide el tipo de aplicación que se va a crear.

Debemos seleccionar **Single Document** (Documento simple) y **View/Document** (soporte de arquitectura Vista/Documento). En los pasos 2 y 3 del asistente dejaremos los valores por defecto. En el paso 4 sólo se seleccionará la opción **3D Controls.** El paso 5 lo dejaremos con los valores por defecto. En el 6 debemos detenernos para cambiar la clase derivada para la Vista, que por defecto se deriva de **CView.** El problema es que esta clase no permite la inserción de controles en su contenedor. Por ello, utilizaremos la clase **CFormView** que sí permite la inclusión de controles.

Con todos estos datos *Visual C++* nos pedirá conformidad para crear el esqueleto principal del programa.

Una aplicación Windows utiliza, al menos, una ventana para que el usuario pueda comunicarse con ella

Ahora ya estamos en disposición de empezar a insertar código en nuestro programa. Para el ejemplo del artículo se ha utilizado como nombre del proyecto **C2,** por lo que la mayoría de los archivos creados comienzan con este nombre seguido de la descripción del archivo. Por ejemplo, para el archivo de Vista se utiliza **C2View.cpp,** y para el del documento **C2Doc.cpp.** El archivo correspondiente a la aplicación se denomina **C2.cpp. MainFrm.cpp** se encarga de crear la ventana principal del programa y todos los controles insertados en la misma.

La funcionalidad del programa será la siguiente: hay una caja de texto con un botón. Cuando se inicie el programa se podrá visualizar dentro de la caja de texto la frase **Mi primer programa....** Si pulsamos sobre el botón, este texto cambiará automáticamente visualizando la frase **"... en Visual C++".** Aunque la funcionalidad del programa no sea espectacular, servirá para comenzar a entender el funcionamiento de los controles y la ventana de un programa en *Visual C++*.

Otro detalle que tiene el programa es que dependiendo del texto que se encuentre en la caja de texto aparecerá visualizado con un tipo de fuente de letra diferente y con una rotación de 45°.

Para comenzar el desarrollo del programa insertaremos en la ventana una caja de texto y un botón de comando en la posición que queramos. Esto se hace desde el editor de recursos *(ResourceView).* Debemos seleccionar la carpeta Dialog y seguidamente el recurso *IDD_C2_FORM,* el cual corresponde al contenedor de controles del formulario de la aplicación. Una vez situados en este recurso, echaremos mano de la barra de controles e insertaremos los dos elementos anteriormente mencionados, manteniendo los valores por defecto.

Viendo las propiedades de cada control, veremos que la caja de

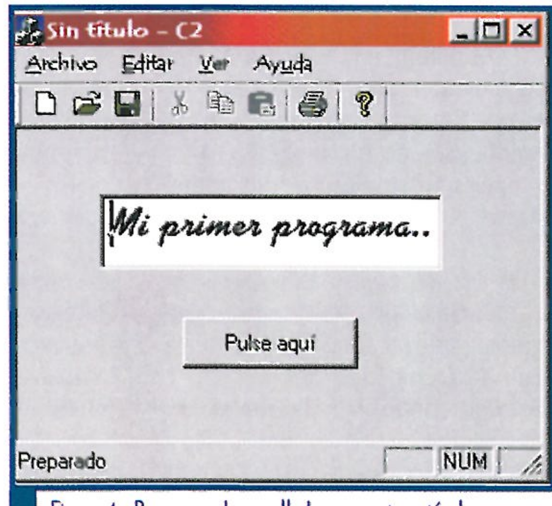


Figura 4.- Programa desarrollado para este artículo.

texto toma el identificador *IDC_EDIT1* y el botón *IDC_BUTTON1.* Estos identificadores nos servirán posteriormente para poder seleccionar cada control dentro del programa.

VARIABLES MIEMBRO

Las variables miembro se utilizan para poder acceder fácilmente a un control sin necesidad de derivar de ninguna clase ni utilizar el identificador asociado. Estas variables pueden definirse del mismo tipo de clase que el control al que representan o como valor de dicho control. Mediante esta última posibilidad, y aplicada a un cuadro de texto, podemos cambiar el contenido del control fácilmente.

Los mensajes de Windows son la base para la ejecución de funciones/procedimientos en nuestros programas

En la Figura 5 podemos ver cómo se define la variable *m_caja* para el control *IDC_EDIT1* insertado en el formulario.

Para definir una variable miembro debemos abrir el asistente de clases. Esto se logra mediante la combinación de teclas *CTRL+W* o la opción *ClassWizard* del menú **View.**

El siguiente paso es seleccionar el identificador del control que vamos a utilizar *(IDC_EDIT1).* Una vez seleccionado, abriremos la pestaña *Member Variables* y pulsaremos sobre el botón **Add Variable.** En este punto se abrirá una ventana en la que podemos determinar el nombre de la variable y su tipo. Para el ejemplo se ha utilizado el nombre **m_caja** con el tipo *CString* para la utilización de cadenas de caracteres (Figura 6).

Esta variable miembro servirá posteriormente para cambiar de forma automática el contenido del texto de la caja incrustada en el formulario.

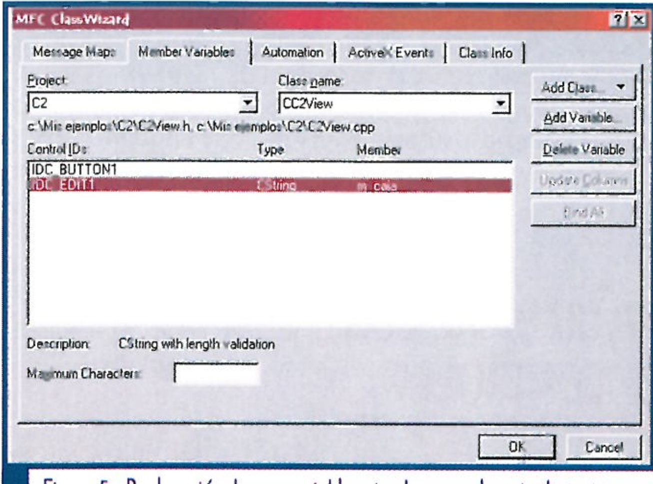


Figura 5.- Declaración de una variable miembro para la caja de texto.

DESARROLLO DEL PROGRAMA

Lo primero que vamos a implementar en el programa es que la ventana quede centrada en relación con la resolución que tenga el escritorio de *Windows.* Para ello debemos abrir el archivo de vista **C2View.cpp** y situarnos dentro del procedimiento *void CC2View::OnInitialUpdate().* Este procedimiento se utiliza cuando se visualiza por primera vez la ventana de la aplicación, por lo que será un buen momento para posicionar la ventana en pantalla.

```
void CC2View::OnInitialUpdate()
{
CFormView::OnInitialUpdate();
GetParentFrame()->RecalcLayout();
ResizeParentToFit();
//Centramos la ventana
GetParent()->CenterWindow ();
//Texto inicial en caja de texto
SetDlgItemText(IDC_EDIT1,"Mi pri-
    mer  programa...");
// Asigna valor a variable miem-
    bro
m_caja="... en Visual C++";
}
```

Como vemos en el código, para centrar la ventana se utiliza el procedimiento *CenterWindow()* aplicado a *GetParent(). GetParent()* nos permite tomar la ventana que contiene nuestra vista (ventana principal del programa). Esto es necesario ya que CenterWindow() debe aplicarse a una ventana determinada. Con una sola línea de código ya tendremos centrada la ventana de nuestra aplicación en relación con la resolución del escritorio de *Windows.*

El siguiente paso será establecer el texto inicial **Mi primer programa...** dentro de la caja de texto. Para ello utilizaremos el procedimiento *SetDlgItemText* de nuestra vista. Este procedimiento tiene dos parámetros:

- Identificador del control a utilizar *(IDC_EDIT1)*
- Cadena de caracteres a visualizar

Podríamos haber utilizado la variable miembro creada anteriormente, pero ésta está pensada para ver la funcionalidad del intercambio de valores en *Visual*



C++, tal y como veremos más adelante.

Para finalizar, asignaremos a la variable miembro **m_caja** la cadena de caracteres **"... en Visual C++"**. Esta cadena se visualizará en la caja de texto sólo cuando se pulse el botón que aparece en la ventana. Esto significa que aunque ahora determinemos un valor para la variable, no se actualizará el contenido de la caja de texto con su valor. Para que eso sea posible, será necesaria la ayuda de otra función que veremos más adelante.

Si ejecutamos ahora el programa veremos nuestra ventana centrada y la caja de texto conteniendo la frase **Mi primera aplicación....** Se puede apreciar que cuando la caja de texto tiene el control el texto está seleccionado de forma automática. Esto se debe a que no hemos restringido la entrada de datos para el usuario, y por lo tanto, el contenido se puede modificar.

Para evitar que se seleccione el texto cuando el control tenga el foco, utilizaremos nuestro primer evento. Además, en él cambiaremos el tipo de fuente de letra para la visualización de caracteres.

Add Member Variable

Member variable name: m_caja

Category: Value

Variable type: CString

Description: CString with length validation

OK | Cancel

Figura 6.- Definición de la variable **m_caja**.

EVENTO PARA LA CAJA DE TEXTO

Para poder crear un nuevo evento debemos abrir el asistente de clases y seleccionar el objeto con el que queremos trabajar *(IDC_EDIT1)*. En la parte derecha del asistente podremos ver los eventos asociados a este control. Nosotros seleccionaremos *EN_SETFOCUS*, que significa "cuando se obtenga el foco para este control...".

Haciendo doble clic sobre este evento aparecerá código de forma automática para que escribamos lo que ocurrirá al ejecutarse este evento. En este momento podremos cambiar la fuente de letra y la selección del texto del control.

Vayamos por partes. Lo primero será crear un objeto derivado de la clase *CEdit* al que posteriormente asignaremos nuestra fuente de letra.

```
CEdit *caja= (CEdit *)
    CC2View::GetDlgItem
    (IDC_EDIT1);
```

El siguiente paso es definir el tipo de fuente de letra que se visualizará. Para crear una fuente de letra es necesario trabajar con la estructura *LOGFONT*. En ella podemos establecer valores de la fuente como itálica, rotación de texto, subrayado, tamaño, etc. Para el ejemplo se ha utilizado la fuente *Kaplan*, aunque el lector podrá utilizar la que desee. Tan sólo será necesario cambiar el nombre de la fuente.

Para la creación de la fuente se utiliza la variable **fuente.** Ésta deberá definirse fuera del procedimiento para que la visualización sea correcta. Un lugar idóneo para la definición es el *header* del archivo de la vista **(C2View.h).** A continuación podemos ver un extracto del archivo conteniendo la definición de la fuente, la cual debe derivarse de la clase *CFont.*

```
...
// Attributes
public:
    CC2Doc* GetDocument();
    CFont fuente;

// Operations
public:
...
```

Para establecer la fuente en la caja utilizaremos el método *SetFont* del objeto caja utilizando como tipo de letra el objeto **fuente.**

```
void CC2View::OnSetfocusEdit1()
{
// Creamos una instancia de CEdit
    // para controlar la caja de
    texto
CEdit *caja= (CEdit *)
    CC2View::GetDlgItem
    (IDC_EDIT1);

// fuente Inicial (Kaplan)
LOGFONT lf;
memset(&lf,0,sizeof(LOGFONT));
strcpy(lf.lfFaceName , "Kaplan");
lf.lfHeight=26;
fuente.CreateFontIndirect (&lf);
caja->SetFont (&fuente);

//Eliminamos la selección
caja->SetSel (0,0);
}
```

Finalmente, para borrar la selección debemos utilizar el método *SetSel* del objeto *caja*. Los dos pará-

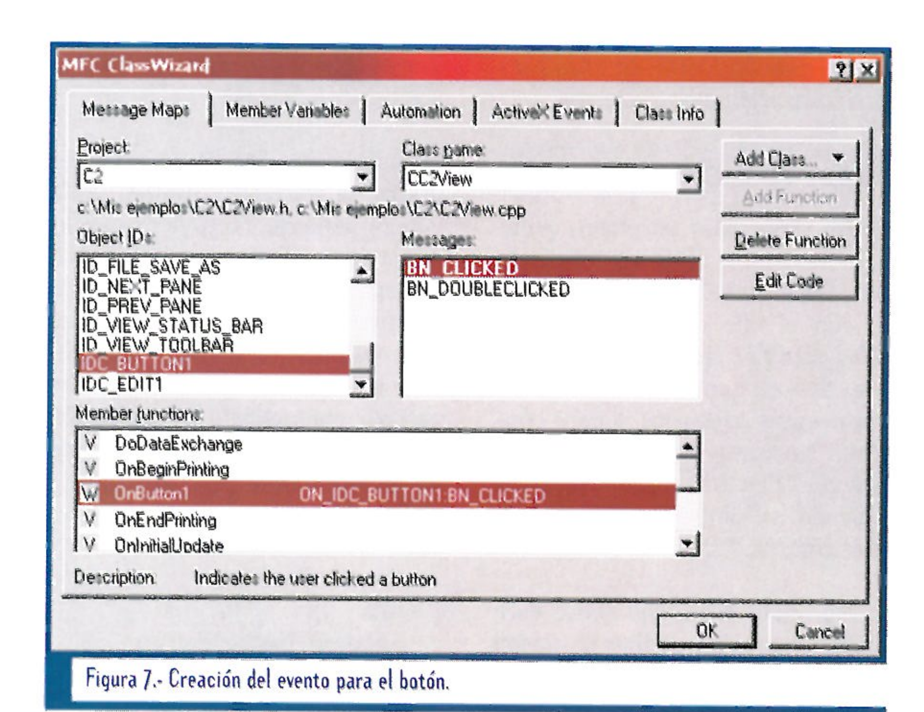


Figura 7.- Creación del evento para el botón.

metros que definen este método representan la posición inicial y final de la selección. Poniendo estos parámetros a cero conseguiremos que no se seleccione ningún carácter.

EVENTO PARA EL BOTÓN DE COMANDO

El botón insertado en el formulario servirá para cambiar el contenido de la caja de texto mostrando una nueva frase con otro tipo de fuente de letra. Para ello, en este caso utilizaremos el valor de la variable miembro **m_caja.**

El sistema de mensajes de Windows hace posible que distintas tareas compartan el procesador

Para la creación del evento utilizaremos *ClassWizard*, seleccionaremos el control *IDC_BUTTON1.* Seguidamente pulsaremos sobre el evento *BN_CLICKED* que permitirá controlar lo que ocurra cuando se pulse el botón de comando (Figura 7).

Nuestra variable miembro puede recibir como valor el contenido de la caja de texto o por el contrario, enviar a la caja de texto el valor de la propia variable. Para que esto sea posible es necesario utilizar el procedimiento *UpdateData(valor).*

Si pasamos como parámetro de este procedimiento el valor **false,** se pasará el contenido de la variable a la caja de texto. El efecto contrario se consigue pasando el valor **true** al procedimiento. Puesto que nosotros queremos actualizar la caja de texto con un nuevo mensaje cuando se pulse el botón utilizaremos el valor **false** como parámetro de *UpdateData.*

Para la nueva fuente utilizada en este procedimiento se ha aplicado una rotación al texto. Para realizarlo, es necesario asignar el valor en ángulos a *lfEscapement.* Ésta es una propiedad de la estructura *LOGFONT* que hemos creado con la variable *lf.*

```
void CC2View::OnButton1()
{
// Creamos una instancia de CEdit
//para controlar la caja de texto
CEdit *caja= (CEdit *)
     CC2View::GetDlgItem
     (IDC_EDIT1);

//Cambiamos el tipo de fuente
     (Comic Sans MS)
LOGFONT lf;
memset(&lf,0,sizeof(LOGFONT));
strcpy(lf.lfFaceName , "Comic
     Sans MS");
lf.lfHeight=26;
lf.lfEscapement =45;

//Establecemos la nueva fuente
fuente.CreateFontIndirect (&lf);
caja->SetFont (&fuente);

UpdateData(false);
}
```

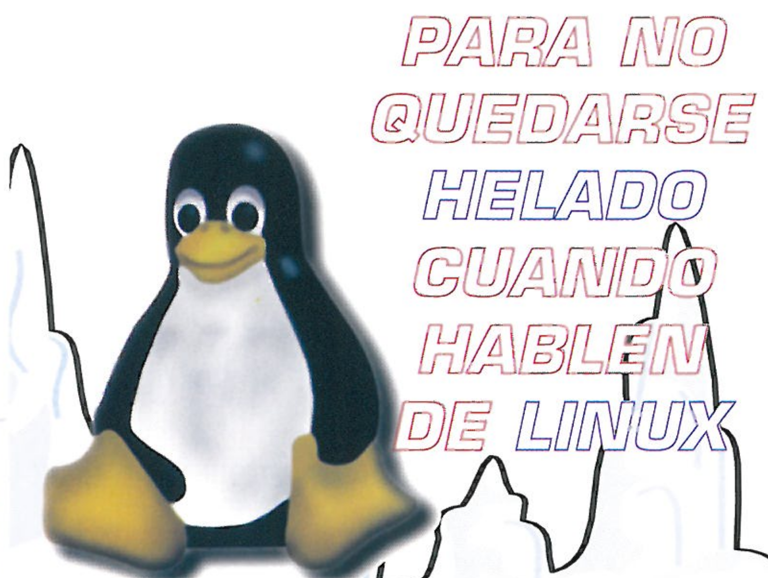
Y con esto hemos completado el desarrollo de nuestra primera aplicación. Cuando se inicia el programa se visualiza un mensaje en la caja de texto con una fuente de letra determinada. Si pulsamos sobre el botón se cambia el texto de la caja utilizando otro tipo de fuente y el efecto especial de rotación de texto a 45 º. Si cambiamos el foco a la caja de texto volverá a visualizarse el texto con la primera fuente utilizada pero el contenido del texto no variará, es decir, seguiremos leyendo **...en Visual C++.**

Emplazo al lector a que experimente cambiando propiedades en los controles utilizados, así como tamaño de fuente de letras, etc.

En el próximo artículo veremos la subclasificación. Se trata de una técnica muy utilizada para, por ejemplo, delimitar los caracteres y valores que podemos introducir en una caja de texto.



QUE NO TE LÍEN CON EL <CÓDIGO>



BOLETÍN DE SUSCRIPCIÓN

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L. (Revista Sólo Programadores).
C/ San Sotero, 5. 1ª Planta. 28037 Madrid. Tlf: 91 304 87 64. Fax: 91 327 13 03

Quiero suscribirme a la revistas SÓLO PROGRAMADORES y SÓLO PROGRAMADORES LINUX desde el Número ....
y beneficiarme de las condiciones de estas magníficas promociones:

❏ **SUSCRIPCIÓN ANUAL**
22 NÚMEROS + 22 CD-ROMs
AL PRECIO DE
14.822 ptas. / 89,09€

Precio de suscripción para el extranjero:
22 NÚMEROS + 22 CD-ROMs
AL PRECIO DE
21.200 Ptas.

FORMAS DE PAGO:

❏ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❏ Transferencia al Banco Popular Español. C/ Valdecanillas, 41.
Nº c/c: 0075/1040/43/ 0600047439
❏ Talón bancario a nombre de REVISTAS PROFESIONALES, S.L
❏ Domiciliación bancaria
❏ Contra reembolso

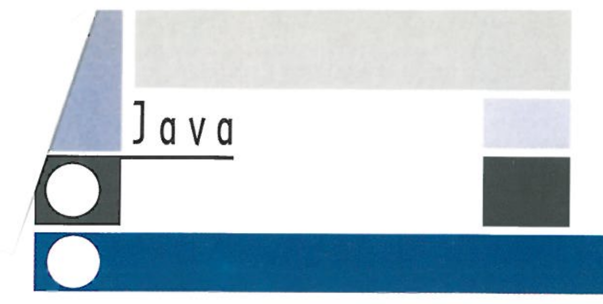
Soy antiguo suscriptor
❏ Sí ❏ No

PARA ENVÍOS AL EXTRANJERO SÓLO SE ADMITIRÁN LAS SIGUIENTES FORMAS DE PAGO:
❏ Giro postal a nombre de: REVISTAS PROFESIONALES, S.L
❏ Talón conformado en dólares a nombre de: REVISTAS PROFESIONALES, S.L
❏ Por VISA ____/____/____/____
Caducidad: __/__

NOMBRE Y APELLIDOS:..
EDAD:............... PROFESIÓN: ....
TFNO: .... DOMICILIO: ....
CIUDAD:...................................... C.P: ....PROVINCIA:..................................

Datos de domiciliación:
Banco..
Domicilio ....
Nº de Cuenta..................|..................|..................|..................
Titular..
Fecha ....
FIRMA

Promoción válida hasta agotar existencias





Firmado de Applets (I)

Javier Sanz Alamillo. *Ingeniero de Software.*

¿Quién no ha cargado una página *Web* en la que se ejecuta un *applet Java* y se ha preguntado si ese *applet* podría realizar acciones peligrosas sobre su ordenador?, y en caso contrario, si yo quisiera que las pudiera hacer, ¿qué tendría que cambiar en el *applet?* Sigue leyendo y lo sabrás.

Muchas páginas en *Internet* incluyen uno o varios *applets*, que no son ni más ni menos que programas *Java* o lo que es lo mismo, un conjunto de clases que se descargan automáticamente desde el servidor *Web* que envía la página, y que se ejecutan en la máquina local que las recibe, bajo el control de un navegador.

APPLETS

Un *applet* ofrece muchas ventajas pero también tiene algunos inconvenientes. Entre las ventajas, se dispone de un programa que se ejecuta a través de un navegador, por lo que basta tener conexión a *Internet* para ejecutarlo, y lo más interesante, desde cualquier lugar, no sólo en la oficina o en una instalación.

Además, se descarga al servidor *Web* de ejecutar esa misma aplicación transformada en muchos *cgi-bin* o *servlets* y se reduce el tráfico posible de red durante la ejecución En resumen, se trata de una aplicación que en último término se ejecuta localmente.

Por otro lado aparecen los inconvenientes. Si el *applet* es muy voluminoso, el descargarse todas las clases puede suponer un tráfico de red inicial elevado y por tanto una cierta espera. Por otra parte, existen ciertos problemas relativos a los navegadores sobre los que se ejecutan, ya que cada uno ejecuta esas clases con ciertas peculiaridades, no muchas, pero algunas sí hay que tenerlas en cuenta.

Aún siguen algunas preguntas sin contestar, ¿puede un *applet* borrar mi disco duro? ¿quién me garantiza la ejecución de ese *applet*?, esto es, ¿alguien me garantiza que el *applet* realiza lo que se prevé?, ¿no será un *applet* una especie de virus que se transmite a través de *Internet*?.

En la Figura 1 se puede observar una típica página *Web* en la se que está ejecutando un sencillo pero eficaz *applet*.

SAND-BOX

Por defecto y por razones evidentes de seguridad, ya que nunca

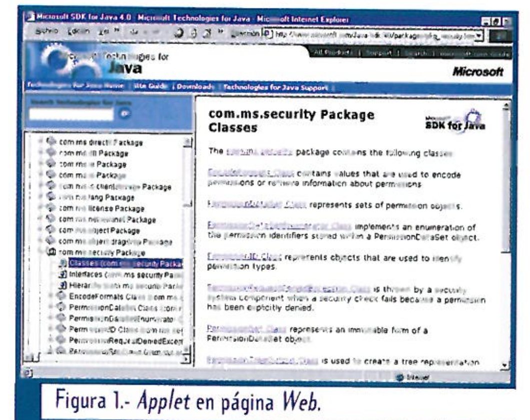


Figura 1.- *Applet* en página *Web*.

se sabe quién ha escrito un determinado *applet*, y lo que es aún peor, con qué intenciones. Como por omisión uno no debería fiarse de la ejecución en su ordenador de programas extraños y ajenos, los *applets Java* están sujetos a unas restricciones de seguridad, lo que se define como que están incluidos en una *sand-box*.

Esto significa simplemente que cuando se diseñaron los *applets* se tuvieron en cuenta toda una serie de posibles problemas de seguridad que podrían ocurrir, por tanto, había que ofrecer una solución para eliminarlos, y cuanto más transparente fuera para el desarrollador mejor.

Entre estas soluciones surgieron las restricciones, las cuales principalmente niegan el acceso a los recursos locales al *applet*. Por ejemplo, un *applet* no puede leer los archivos locales de la máquina sobre la que se ejecuta, ni borrarlos, ni crearlos.

Tampoco puede abrir *sockets* para comunicarse con otras máquinas, ni averiguar determinada información de su sistema. En resumen, garantizar que el *applet* nunca pueda atacar la máquina sobre la que se ejecuta. Si un *applet* no puede realizar nada sobre la máquina local y yo quiero desarrollar una aplicación que requiere de ciertos parámetros del sistema o copiar ciertas cosas en esa

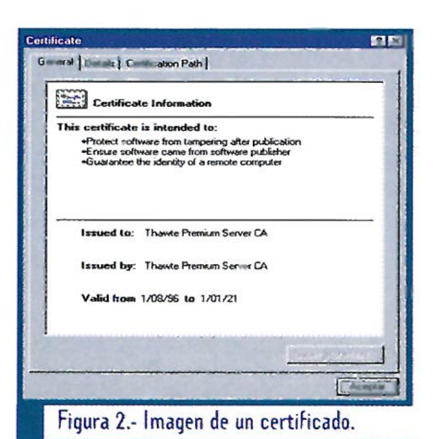


Figura 2.- Imagen de un certificado.

máquina, ¿cómo se debe efectuar?

Gracias a la sand-box los applets no pueden realizar actividades peligrosas



Figura 3.- Instalación de certificado.

Los primeros navegadores disponibles no tenían prevista esta circunstancia, lo que significaba que los *applets* no podían realizar nada fuera de la *sand-box*, y comenzó a ser un problema cuando muchas aplicaciones *Java* pasaron de ser los típicos *Frame* a *applet* para que se pudieran ejecutar remotamente a través de un navegador.

Esta circunstancia había que solucionarla rápidamente, pero teniendo en cuenta que se trataba de una cuestión muy delicada y que la solución debía ser robusta y sin fallos.

Por ello, únicamente los navegadores más "recientes" están preparados (a partir de *IE4* en *Windows* y *Navigator o Communicator 4* en todas sus plataformas disponibles) para solucionar esta eventualidad. El problema aparece debido a que cada navegador soluciona este problema de una manera diferente, por lo que el problema ya no es solamente el *applet* en sí, sino el navegador sobre el que se ejecuta.

La solución adaptada para resolver el problema es común en los navegadores, aunque lo que es diferente es la forma de resolverlo y todos se basan en que los *applets* deben estar *firmados digitalmente*.

Para ello, se debe generar y aplicar sobre los *applets* un certificado digital, que es único e irrepetible, y que relaciona al *applet* con el desarrollador del mismo. De esta manera si el *applet* realiza alguna tarea fuera de lo común, sería posible reclamar al autor. Además, con este sistema se ejecutarían sólo los *applets* de determinas entidades, aquellos que ofrezcan cierta confianza.

CERTIFICADOS

Entonces, para poder realizar determinadas tareas con un *applet* se debe firmar digitalmente mediante un certificado.

De un modo sencillo, se puede decir que un certificado es un conjunto de información (una firma digital) que proviene de una entidad certificadora que garantiza que ese certificado corresponde a la persona que dice que es, y por tanto, responde de él.

Los certificados emitidos por autoridades certificadoras ofrecen todas las garantías a los usuarios

Brevemente, un certificado consta de dos partes, una clave pública y una clave privada. La clave privada es lo que se utilizada

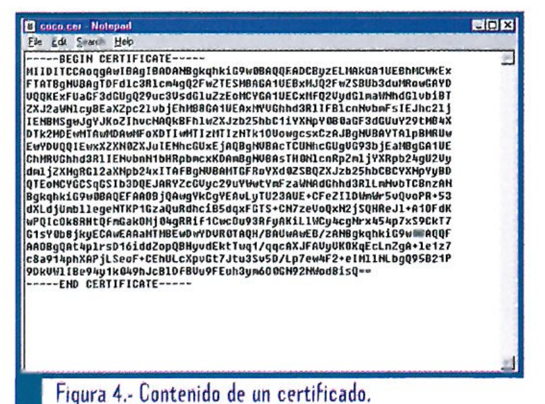


Figura 4.- Contenido de un certificado.

para firmar el *applet*, el código propiamente dicho. La clave pública es lo que se aplicará en la máquina que ejecutará el *applet* para verificar que el código fue firmado con esa clave privada, de tal forma, que si durante el proceso no hay relación entre las claves, no es posible ejecutar el código.

En la Figura 4 puede observarse el contenido de un certificado real.

AUTORIDADES DE CERTIFICACIÓN

Las autoridades de certificación son entidades en las que se debe confiar, y que expiden, mejor dicho, venden certificados, porque no son gratis. ¿Pero qué define que una autoridad de certificación tenga mi confianza? Este problema se ha solucionado de una forma relativamente sencilla. Cada fabricante de navegadores incluye en los mismos un certificado de cada autoridad certificadora válida, lo cual le permite responder de cada certificado que distribuye.

En otras palabras, cada navegador entiende un determinado tipo de certificado, por lo que los *applets* firmados constan de dos certificados realmente, el que se usa para las tareas propias de firmar el certificado y el segundo que garantiza que el primer certificado es correcto.

Existen muchas autoridades de certificación, de entre las que destacan por ejemplo por su calidad y rapidez de servicio, *Verisign*, http://www.verisign.com, que suele cobrar unos $400 por un certificado clase 2 para los navegadores *Netscape* e *Internet Explorer*. Conviene comentar que generalmente los certificados tiene una validez de un año.

Otra entidad renombrada, más que nada porque ofrece gratuitamente certificados por un período de un mes, es *Thawte Certificacion*, http://www.thawte.com, que para sorpresa de muchos vende un certificado clase 2 para navegadores *Netscape* y *Explorer* por la cifra de $200. Es lo bueno de la competencia.

Otras entidades más o menos reconocidas por su uso son: *SECUDE*, http://www.secude.com, *Entrust* http://www.entrust.com y *Entropia CA*, http://www.entropia.com.

Existe la posibilidad de no utilizar una autoridad certificadora como las anteriores y convertirse uno mismo en su propia entidad. Para ello, basta con comprar el *software* que genera certificados y comenzar a establecer relaciones con los fabricantes de navegadores.

También existe la posibilidad de crear uno sus propios certificados de prueba. Existen muy buenas herramientas, y pueden ser descargadas de las *web* corporativas de *Netscape* o *Microsoft*. Más adelante se mostrará cómo realizar este proceso.

EL PROCESO DE FIRMAR APPLETS

En función del navegador, el proceso de firma de un *applet* presenta algunas variaciones. Si se compara el proceso que se debe realizar entre los navegadores *Netscape (Navigator o Communicator)* e *Internet Explorer*, en éste último todo es más sencillo de llevar a cabo y permite realizar una nueva serie de tareas sobre los *applet*s. Por ejemplo, implementar un sistema de instalación de clases en local, que hacen que la mayoría de las veces los desarrolladores se decanten a firmar *applet*s casi exclusivamente para *Internet Explorer*.

Aparte de esto, el proceso de firma no afecta prácticamente al código del mismo *applet* si el programador no quiere complicarse mucho en el desarrollo. El proceso de firmar un *applet* con *Internet Explorer* pasa prácticamente por firmar un fichero *cab* que contiene el *applet* y las clases con las que se relacione.

Cuando el archivo es cargado por el *Explorer*, éste pregunta inmediatamente al usuario si confía en él, mostrándole información sobre el creador, permisos que solicita, etc. Si se contesta afirmativamente, el *applet* comienza a ejecutarse con los permisos requeridos. Si se deniega, el navegador intenta cargar las clases que componen el *applet* pero las ejecuta dentro de la *sand-box*, o sea, con las restricciones comunes, por lo que la mayoría de las veces no funcionará.

El problema de firmado de *applets* al utilizar *Netscape* se refiere a la compatibilidad entre las diferentes plataformas en las que existe (*Unix, Windows, Mac*), ya que obliga al programador a modificar el

código del *applet*, definiendo en cada caso el proceso que se debe seguir para ejecutar determinadas acciones prohibidas.

Las diferencias entre navegadores hacen que la tarea de firmar un applet se complique

Para permitir este tipo de programación está definida una *API* con unas clases especificas para *Netscape*, el denominado *Netscape Capabilities API*. Conviene comentar que esta *API* ha sido una de las más estudiadas por mucha gente, puesto que nada más salir una versión de navegador para *Netscape*, comenzaba una carrera en búsqueda del error de seguridad de turno, descompilando y estudiando miles y miles de clases.

Volviendo al método que se utiliza en *Netscape*, cuando un *applet* es cargado se le aplican directamente las restricciones de seguridad por defecto, y cuando el *applet* intenta realizar una acción prohibida se comprueba si el *applet* es firmado y se pregunta al usuario, si permite o no su ejecución.

Si la deniega, esa acción no se ejecuta, pero el *applet* continúa ejecutándose. Si se permite, la acción se ejecuta y a partir de ese momento se le considera como amigo y puede realizar tareas específicas consideradas "peligrosas".

Otra característica que diferencia a estos navegadores es el formato en el que se empaqueta el *applet* firmado. En el caso de *Netscape*, el *applet* se firma utilizando una herramienta denominada *Netscape Object Signing ID.* Se encarga de crear un fichero "*manifiest*" y las clases del *applet,* generando un fichero *JAR* (*Java ARchive*) que es gestionado por el navegador.

En el caso de utilizar *Internet Explorer*, el fichero que contenga las clases debe ser un fichero tipo *cab*, como se comentó anteriormente, y utilizar una herramienta que *Microsoft* ofrece gratuitamente, que se denomina *signtool*, lo que genera un *Microsoft Authentico* de *ID*, que permite el proceso de firmado.

Anteriormente cuando se detallaron las autoridades certificadoras se indicó que los certificados suelen tener una validez de un año, con lo que surge la pregunta, ¿qué pasa con ese *applet* firmado cuando el certificado deja de tener validez, o sea, caduca?, ¿el *applet* sigue siendo válido?

Pues para no variar *Microsoft* y *Netscape* hacen cosas diferentes. Las herramientas de *Netscape* no dejan firmar un *applet* cuyo certificado ha caducado, aunque para sorpresa de todos, si se ejecuta *applet* ya firmado con un certificado caducado, este *applet* continuará ejecutándose como si todo estuviera correctamente, como si tuviera un certificado válido. ¿Quién actualiza el certificado anualmente entonces?

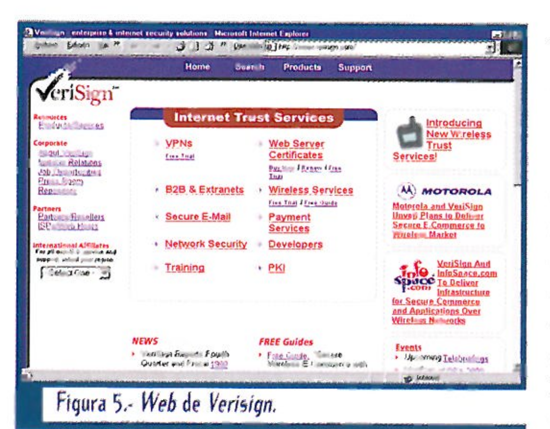


Figura 5.- Web de Verisign.

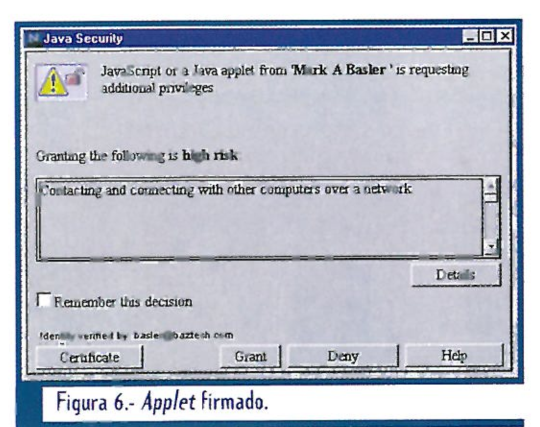


Figura 6.- *Applet* firmado.

Microsoft por su parte, para firmar un *applet* exige un proceso adicional que se denomina *timestamped*. Aunque se detallará más adelante, *Internet Explorer* comprueba el firmado y el *timestamped* y si son correctos trata el *applet* como seguro.

Si el *applet* no tiene el *timestamped* o si la fecha de éste supera la fecha de validez del certificado, el *applet* se considera como no seguro y se le aplican las restricciones de seguridad que aplica la *sand-box*.

VERSIONES DE NAVEGADORES

Hoy en día la mayoría de los usuarios de entornos *Windows* disponen de versiones actualizadas de *Communicator* o *Internet Explorer* actuales, 4.6 para *Netscape*, 5.0 para *Internet Explorer*. Si no disponen de una versión actualizada, podrá encontrarlas en el *CD-ROM* de la revista.

Esta misma circunstancia no ocurre en los entornos *Unix*, en los que la actualización del navegador de *Netscape* es algo más lenta, y no en todos los sitios se dispone o está instalada. Por este motivo se suele definir la siguiente relación para garantizar que los *applet* firmados funcionarán correctamente:

- Utilizar *Netscape* versión 4 o superior.
- *Microsoft Internet Explorer* versión 4.0 o superior bajo *Windows*.

Comentar que *Internet Explorer* 4.0 en *Mac* no reconoce los *applets* firmados. De todas formas, con la rapidez actual no es de esperar que el desarrollador se encuentre que un *applet* firmado produzca errores por utilizarse en un navegador no válido.

FIRMANDO APPLETS PARA INTERNET EXPLORER

Normalmente, el desarrollador apenas necesita modificar el código de su *applet* para que éste sea firmado y funcione, lo cual es muy práctico e interesante. Además, evita posibles problemas de corrección de errores, ya que no hay que controlar errores en el código que ejecuta acciones prohibidas.

De todos modos, existe una circunstancia por la cual es necesario escribir código aparte del propio de "negocio" y para ello se utiliza una *API* de seguridad propia de *Microsoft*, el paquete denominado *com.ms.security*. Resulta necesario si el *applet* tiene que realizar tareas prohibidas en los métodos *init(), start(), stop() o destroy()*.

Por ejemplo, si el *applet* tiene que leer un fichero del disco local para inicializarse, el lugar ideal sería *init*() o mejor *start*(). Si realiza esto, el *Explorer* mostrará una maravillosa excepción *SecurityExceptionEx[Host]* que le deja a uno asustado. Esto es debido a que el proceso para comprobar si una acción se puede o no llevar a cabo lo determina la máquina virtual *Microsoft VM* (*Virtual Machine*) que incluye el navegador.

El firmado de applets es muy sencillo si se utiliza Internet Explorer

Cuando un *applet* firmado tiene que realizar una tarea prohibida, la *VM* comprueba si el método tiene permisos para realizar la acción o por el contrario no debería ejecutarla. Por defecto, estos métodos no tienen estos permisos.

Un método tiene permisos o es *trusted* si utiliza el método *assertPermission*() de la clase *PolicyEngine* en el paquete de seguridad *com.ms.security*. Mediante esa invocación, el *applet* solicita a la *VM* la posibilidad de realizar una tarea prohibida. Por el contrario, se dice que un método es *untrusted* si es llamado desde un método *init(), start(), stop() o destroy()*.

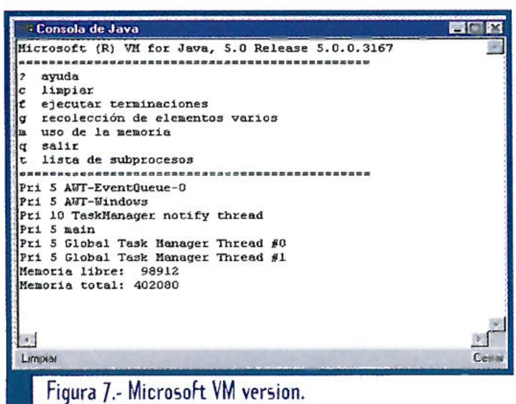


Figura 7.- Microsoft VM version.

TABLA 1. Tipos de permisos.

PermissionID.EXEC	Permiso para ejecutar Aplicaciones
PermissionID.FILEIO	Permisos para control y realización de operaciones de E/S
PermissionID.MULTIMEDIA	Control sobre la clase com.ms.security.permission. MultimediaPermission
PermissionID.NETIO	Control de operaciones de red
PermissionID.PRINTING	Control de operaciones de impresión
PermissionID.PROPERTY	Acceso a las propiedades del sistema
PermissionID.REFLECTION	Control sobre el uso de la API de Reflection
PermissionID.REGISTRY	Control de acceso al registro del sistema
PermissionID.SECURITY	Control del acceso a la API de seguridad del JDK
PermissionID.SYSSTREAMS	Control de los streams del sistema
PermissionID.SYSTEM	Permiso que representa todos los permisos
PermissionID.THREAD	Control de operaciones sobre threads
PermissionID.UI	Control para acceder a funcionalidades de AWT
PermissionID.USERFILEIO	Control de acceso a operaciones de E/S sobre ficheros

De todos modos, hay una forma de evitar el uso de las clases *Microsoft*, pues que en caso de utilizarlas, habría que incluirlas en la descarga de clases necesarias para ejecutar el *applet*. La solución es muy sencilla, en vez de utilizar los métodos propios del *applet init(), start(), stop() o destroy()* para ejecutar las tareas prohibidas, crear un *thread* y realizar en éste las tareas problemáticas.

DESARROLLO DE UN MÉTODO "TRUSTED"

Para realizarlo, debemos utilizar la clase *PolicyEngine* e invocar el método *assertion* que tiene la forma:

```
public static native void asser-
    tion (PermissionId pid),
```

donde *PermissionId pid* es el tipo de permiso que se pretende conseguir.

Los tipos de permisos son de lo más variados y se pueden encontrar en la siguiente Tabla 1:

Veamos un ejemplo que aclarará todo el proceso. Para que el ejemplo funcione, debe ser incluido en un *applet* firmado como se mostrará seguidamente o ejecutarse en *Microsoft Developer Studio* o puesto en el *classpath* del sistema local.

Lo más importante es lo que se sale fuera de lo normal en el desarrollo de un *applet*. Se observa que se ha hecho un *import* de la *API* de seguridad de *Microsoft*, luego ya tenemos que instalar localmente o descargar ese paquete.

Se realiza una llamada el método:

```
PolicyEngine.assertPermission (
    PermissionID.SYSTEM);
```

Con lo que conseguimos un control total sobre los recursos del sistema local, al aplicar *PermissionID. SYSTEM*.

A partir de ese momento, se puede realizar cualquier actividad prohibida, como obtener propiedades del sistema o escribir un archivo. Como se observa, es sencillo, pero se tiene que modificar el código del *applets*.

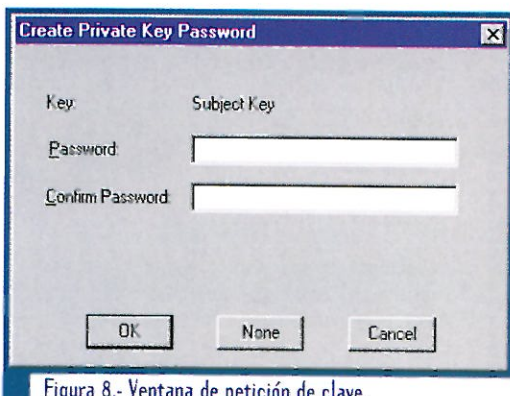


Figura 8.- Ventana de petición de clave.

Aunque no es sólo eso, más adelante se reescribirá ese ejemplo para poder ser utilizado con *Netscape*.

Vamos a describir el proceso de firmado para cualquier *applet* y se lo aplicaremos al código anteriormente descrito.

PROCESO DE FIRMADO

Como se ha venido describiendo hasta ahora, para realizar el firmado de un *applet* con *Explorer* es necesario disponer de una versión de navegador superior a la 4.0, un cer-

LISTADO 1. Ejemplo método trusted.

```
import java.applet.*;
import java.awt.*;
import java.io.*;
import com.ms.security.*;

public class crearFich extends Applet
  implements Runnable {
    Thread hilo;
    Graphics g;
    String CLASE = "com.ms.security.PolicyEngine";

    public void init() {
          g = this.getGraphics();
          hilo = new Thread ( this );
          hilo.start();
    }

    public void run() {
          try {
                if ( Class.forName(CLASE) != null ) {
PolicyEngine.assertPermission ( PermissionID.SYSTEM);

g.drawString ( "capturando ..",10,10);
String userdir= System.getProperty("user.dir");
g.drawString ( "leida propiedad", 10,50);

String fich = "tmpfoo";

DataOutputStream dos;
dos = new DataOutputStream (
new FileOutputStream ( fich ));

                        dos.writeChars ("Escribiendo cars.");
                        dos.close();
                        g.drawString ("He escrito " + fich, 10,90);
                }
          }
          catch ( Exception e ) {
                g.drawString ( "crearFich Exception " + e, 10, 90);
          }
    }
}
```

tificado compatible con el navegador y un conjunto de herramientas como **cabarc**, **signcode**, **signer.dll**, **java-sign.dll** *y* **chktrust** que se encuentran con la instalación del *Microsoft SDK for Java 2.02* o superior.

La obtención del *Internet Explorer* no será un problema para nadie, pero quizás no está tan claro como obtener un certificado, y más aún, gratuitamente. En la *Web* de *Verisign* se puede comprar un certificado, previo pago de una cantidad importante, aproximada de $20 año para usuarios individuales.

Una vez que se recibe el certificado, se dispondrá de dos ficheros, el certificado propiamente dicho, un fichero con extensión *.SPC (Software Publishing Certificate file)* y la clave privada, con extensión *.PVK* (*private key file*) necesarios al utilizar las herramientas de firmado.

Se puede obtener un certificado gratuito de prueba del *Web* de *Thawte*, que se instala directamente en el registro de *Windows* sin necesidad de ficheros *.SPC* o *.PVK*, lo que obliga a realizar ciertos cambios a la hora de utilizar las herramientas que tratan los certificados.

De todos modos, y para nuestra alegría, podemos crear nuevos propios certificados para realizar las pruebas que requeramos hasta decidir si se compra o no un certificado real.

HERRAMIENTAS BÁSICAS A UTILIZAR

Una vez instalado el *SDK* para *Java* disponemos de una serie de herramientas que utilizaremos a lo largo del proceso de firmado. Se encuentran en el directorio *<instalacion_SDK>\bin\PackSign*. Destacan **cabarc.exe**, un generador de archivos *.CAB* o archivos de compresión, algo así como los *JAR* de *Microsoft*. También **makecert.exe**, que crea certificados, **signcode.exe** que firma el fichero *.CAB* y *chktrust.exe* que comprueba su validez.

CREANDO UN CERTIFICADO

Para crear un certificado personal, se siguen los siguientes pasos:

- Utilizar la herramienta **makecert.exe** para generar el certificado. Por ejemplo, se ejecuta:

```
makecert /sv "ms_claprv.pvk"
            /n
   "CN=Certificado de prueba"
   ms_cert.cer
```

Con ello obtenemos una clave privada, **ms_claprv.pvk**, un fiche-

ro que se utilizará para crear el certificado propiamente dicho, **ms_cert.cer** y que nombraremos como *Certificado de prueba*.

En este proceso aparecerá una ventana, Figura 8, en el que se pedirá la clave para proteger el certificado y la clave privada, proceso que se repetirá en dos ocasiones.

- Generar el certificado, mediante **cert2spc.exe,** que se realiza de la forma siguiente:

```
cert2spc ms_cert.cer ms_cert.spc
```

Así se genera un fichero *.SPC* que es el certificado.

- Indicar al *Explorer* que acepte el certificado. Este paso es diferente si se utiliza un certificado generado o si se compra a una autoridad certificadora.

Si el certificado es comprado, por ejemplo uno de *Thawte*, éste debe ser instalado en *Internet Explorer*. Para ello, seleccione el archivo, éste ya es reconocido como del tipo "Certificado Seguridad" y pulse el botón de la derecha. Aparecerá la opción de **Instalar Certificado.** Se selecciona y se arranca un asistente que permite la importación de certificados. Siguiendo las instrucciones de este asistente se consigue fácilmente la instalación.

Mediante unos sencillos pasos se puede crear un certificado personal

Puede comprobar la lista de certificados instalados mediante las opciones del navegador, **Herramientas->Opciones Internet->Pestaña contenido->Certificados.**

Como tenemos un certificado generado, hay que indicar que es un *"test root"*. Un *test root* es un certificado que sólo se aplicará para tareas de *"debug"* y normalmente no es utilizable (al menos que alguien determine su validez). Para realizar esta tarea y hacer que nuestro certificado tenga total validez hay que ejecutar el siguiente comando que viene con el *SDK* para *Java*:

```
setreg 1 TRUE
```

El problema de esto es que debe de repetir esta tarea por cada máquina que desee que acepte ese certificado. También conviene indicar que una vez ejecutado este comando, cualquier otro certificado generado tendrá validez, por lo puede llegar a ser un problema de seguridad, así que seguramente deseará deshabilitar esta posibilidad. Para ello no tiene más que ejecutar:

```
setreg 1 FALSE
```

y el problema queda resuelto.

A partir de este momento, con los dos ficheros generados, y mediante unos sencillos pasos más, podrá firmar cualquier *applet* para *Microsoft Explorer*.

En el próximo artículo se completará el firmado de *applet* utilizando *Internet Explorer*, se explicará además cómo realizar esa tarea utilizando *Netscape* y se mostrarán diferentes aplicaciones del firmado, como la posibilidad de instalación permanente de paquetes como medio de distribución de clases con el *Explorer*.

CONCLUSIONES

A lo largo de este artículo se ha presentado el concepto de *applet* firmado, mostrando todo el conjunto de nuevas posibilidades que ofrece, eso sí, siempre que el usuario decida que puede realizar algo fuera de la *sand-box*.

También se han descrito los certificados, las autoridades de certificación y los diferentes escenarios en el firmado de un *applet*, basándonos principalmente en los navegadores más utilizados, *Navigator* o *Communicator e Internet Explorer*.

Además, se ha mostrado cómo disponer de certificados de pago o cómo crear certificados personales, así como las herramientas disponibles para gestionarlos.

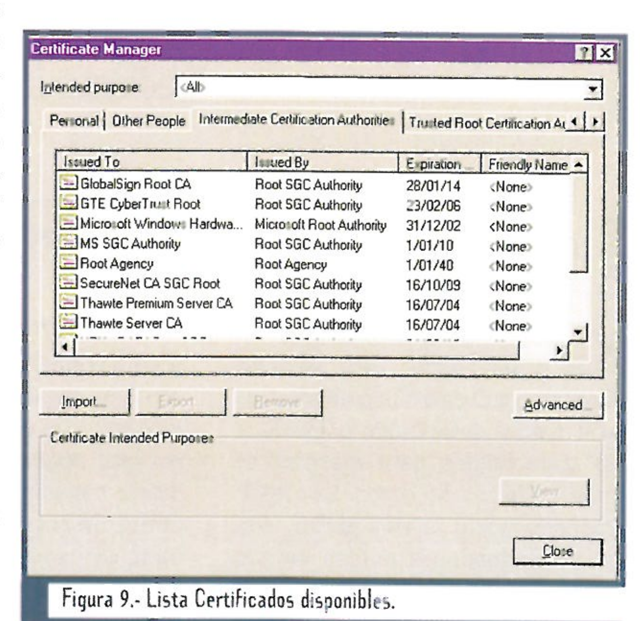
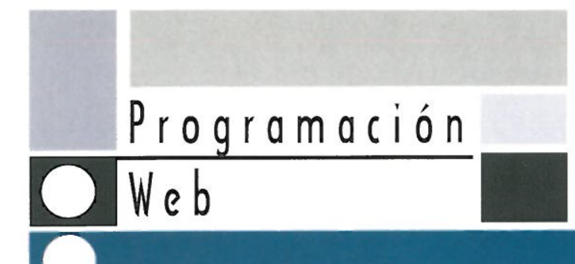
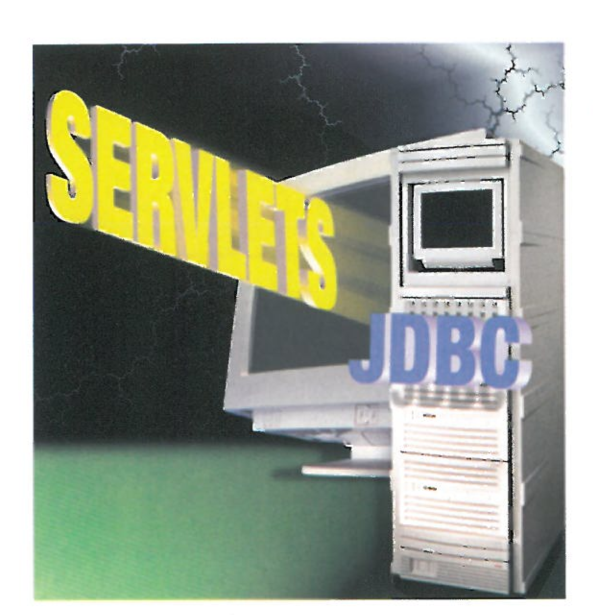


Figura 9.- Lista Certificados disponibles.





Aplicaciones Web con acceso a BBDD basadas en Java (III)

Adolfo Aladro García.
Analista Programador

Es el momento de tomar contacto con la *API JDBC* de *Java,* además de adentrarnos en el proceso de integración con *servlets*. El resultado presentará una aplicación *Web* donde haremos peticiones desde una página *Web* a un servlet que se comunica a su vez mediante *JDBC* con una *BBDD.*

ACCESO A BBDD CON JAVA

Desde que apareció por primera vez la plataforma *Java* se tuvo claro que su desarrollo como tecnología estaba íntimamente ligado a sus capacidades para integrar el acceso a bases de datos. Con este propósito nació la *API JDBC*. Así los programadores podían contar con un mecanismo que les permitiría acceder a bases de datos locales y remotas mediante una interfaz común e independiente de la plataforma.

La *API JDBC* está constituida por un conjunto de clases e interfaces *Java,* contenidas en el paquete *java.sql.* Con estas clases podemos realizar conexiones con bases de datos, consultas *SQL,* definir conjuntos de resultados, etc. En resumen, el uso de esta *API* permite conectar con una base de datos, realizar consultas *SQL* y procesar los resultados para presentarlos después de la manera que resulte más conveniente.

CONTROLADORES JDBC

Un controlador JDBC se define como una librería de software que permite acceder a una deter-



minada fuente de datos. Cada sistema de administración de bases de datos requiere un controlador diferente. Para trabajar con *JDBC* se precisa tener controladores que permitan acceder a las distintas bases de datos. Éstos no son más que un conjunto de clases *Java* que implementan interfaces *JDBC*, que son simplemente colecciones de métodos abstractos sin cuerpo. Los controladores *JDBC* se pueden clasificar en cuatro tipos distintos.

CONTROLADORES PUENTES JDBC-ODBC

Estos controladores simplemente traducen las llamadas *JDBC* a llamadas equivalentes *ODBC*. Las siglas *ODBC (Open Database Connectivity)* responden a un protocolo estándar para el acceso a datos proporcionados por servidores de bases de datos *SQL*, como por ejemplo *Microsoft SQL Server*.

De lo anterior se deduce que en una arquitectura cliente/servidor como la que representa *Internet*, se precisa de la existencia de archivos binarios que, para usar este tipo de controladores, implementen el protocolo *ODBC* en el cliente si estamos considerando, por ejemplo, el modelo en dos capas del que hablamos en el primer capítulo.

En general, el uso de este tipo de controladores implica renunciar a la independencia con respecto a la plataforma y, por otro lado, constituye la solución menos eficaz ya que solamente estamos añadiendo otra capa más de *software* a la estructura existente de acceso a bases de datos mediante *ODBC*.

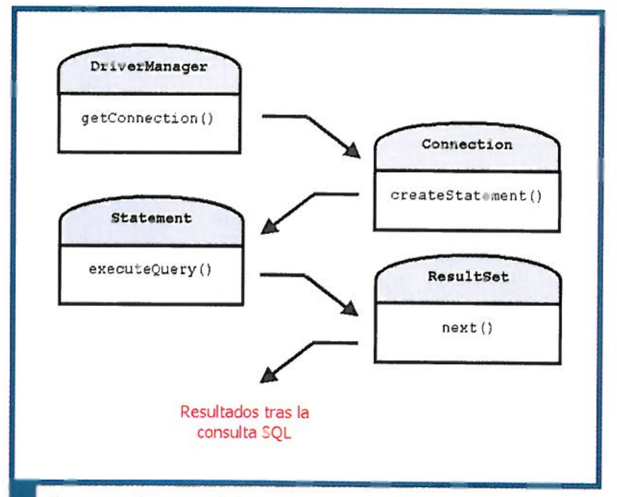


Figura 2.- Resumen de los pasos que se deberán seguir para acceder a una base de datos y lanzar una consulta *SQL* con *JDBC*.

La API JDBC está constituida por un conjunto de clases e interfaces Java, contenidas en el paquete java.sql

Sin embargo esta opción puede ser adecuada para integrar *Java* en sistemas ya desarrollados. Además también es muy útil desde un punto de vista didáctico ya que todos aquellos que deseen conocer la *API JDBC* pueden empezar a estudiarla en su ordenador personal sin tener nada más que el propio *Windows* y una base de datos hecha, por ejemplo, en *Access*.

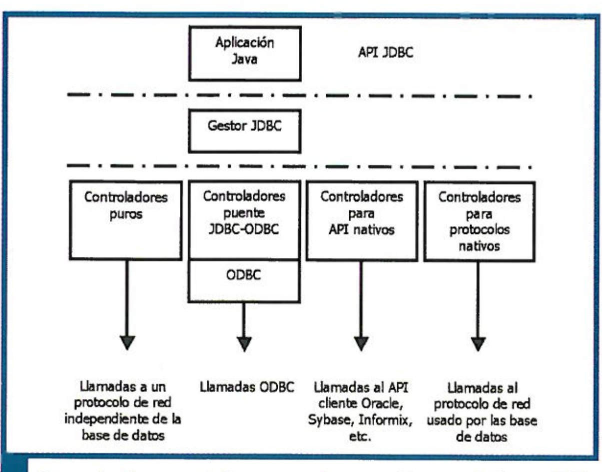


Figura 1.- Esquema de los principales tipos de controladores *JDBC*.

CONTROLADORES PARA API NATIVOS

Este tipo de controladores convierten las llamadas *JDBC* en llamadas a la *API* cliente para *Oracle, Sybase, Informix, DB2* y otras bases de datos.

CONTROLADORES PARA PROTOCOLOS DE RED INDEPENDIENTES

Traducen las llamadas *JDBC* a un protocolo de red independiente de las bases de datos. Éste a su vez será traducido a protocolos nativos, determinados en cada caso por el fabricante de la base de datos al que se pretende acceder. Su función es conectar a los clientes *Java* con diferentes bases de datos. Se trata del enfoque más flexible.

CONTROLADORES PARA PROTOCOLOS DE RED NATIVOS

Convierten las llamadas *JDBC* al protocolo de red usado directamente por las bases de datos, permitiendo, en *intranets*, el acceso directo desde clientes a bases de datos en servidores dados. Se trata, en definitiva, de un caso especial del controlador anterior en el que la comunicación se establece directamente con el servidor de la base de datos mediante un protocolo usualmente propietario. Por ello, los fabricantes de bases de datos son los que deben proveer este tipo de controladores.

DIRECCIONES URL JDBC

Cuando se pensó por primera vez en la forma en la que se podía hacer referencia a una base de datos desde el código *JDBC* se llegó a la conclusión de que dicha forma debía cumplir los siguientes requisitos:

- Diferentes controladores pueden utilizar diferentes formas de referirse a las bases de datos. Por ejemplo, si se utiliza el controlador puente *JDBC-ODBC* solamente se necesita el nombre la fuente de datos. En cambio, si nos encontramos en un entorno de red la forma de especificar la base de datos tiene que permitir indicar datos tales como el ordenador de la red donde se encuentra la máquina de la base de datos así como el puerto de acceso.
- Si una usuario descarga por ejemplo un *applet* que se comunica con una base de datos sería preciso que la conexión fuera posible sin que el usuario tuviera que hacer ninguna tarea de administración de sistemas. Esto significa que toda la información necesaria para establecer la conexión ha de estar contenida en la propia forma de referirse a la base de datos.

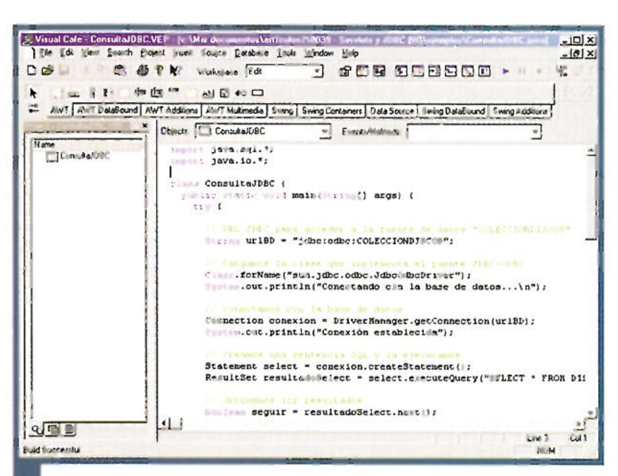


Figura 4.- Código fuente de la aplicación *Java* que realiza una consulta simple a la base de datos.

Un controlador JDBC se define como una librería de software que permite acceder a una determinada fuente de datos

- Finalmente, sería deseable que la referencia a una base de datos pudiera mantener distintos niveles de indirección de forma que el primer nombre pudiera resolverse por medio de algún sistema de nombres de red. De esta forma se evitaría la presencia de nombres de ordenadores privados en la referencia a la base de datos.

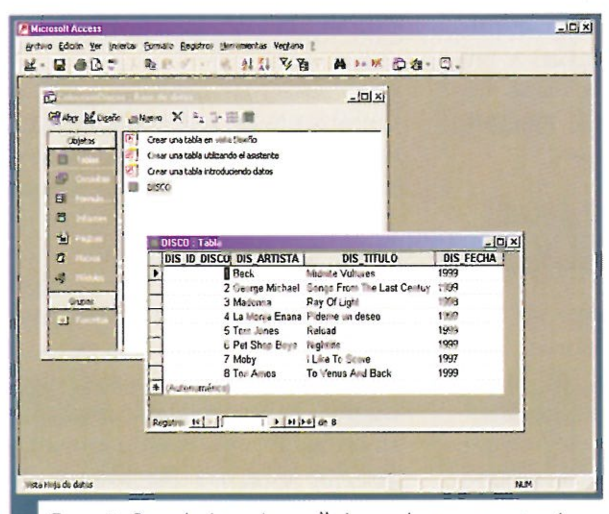


Figura 3.- Base de datos desarrollada con *Access* que actuará como fuente de datos.

Todas estos requisitos son cumplidos por las direcciones *URL* que habitualmente manejamos en la Red. Por ello, la *API JDBC* utiliza lo que denominamos direcciones *URL JDBC*. Una dirección *URL JDBC* tiene tres partes:

```
jdbc:<subprotocolo>:<nombre>
```

- Protocolo o modo de acceso.
- Subprotocolo: Indica una forma determinada de conectarse a una base de datos que puede ser soportada por uno o más controladores.
- Nombre: La sintaxis y el contenido de esta parte de la dirección *URL JDBC* está íntimamente ligada al subprotocolo definido en la anterior.

Por ejemplo:

```
jdbc:odbc:COLECCIONDISCOS
```

El subprotocolo de acceso a bases de datos también puede incorporar el nombre del ordenador de la base de datos, quedando, por ejemplo, como:

```
jdbc:odbc://www.miweb.es
    /COLECCIONDISCOS
```

También podría indicarse un servicio de nombres, lo que proporciona un nivel adicional de indirección.

En general la sintaxis de la dirección *URL JDBC* depende de la arquitectura donde se desenvuelva la aplicación así como del sistema gestor de base de datos. Por ello, antes de nada es recomendable



consultar la documentación de la misma.

EL GESTOR DE CONTROLADORES JDBC

El gestor de controladores *JDBC* es la capa de software que se encarga de gestionar todos los controladores *JDBC* del sistema. Establece una correspondencia entre la base de datos y la dirección *URL JDBC* que se utiliza como referencia. A esta tarea se le denomina registrar un controlador y para realizarla el gestor necesita saber, como es evidente, qué controladores tiene y dónde están.

El método *forName* sirve a este propósito y en definitiva permite cargar la clase *Java* inicial que se corresponde con el controlador *JDBC*. Por ejemplo, para el controlador puente *JDBC-ODBC:*

```
Class.forName("sun.jdbc.odbc.Jdb-
    cOdbcDriver");
```

Nótese que la instrucción anterior debe ser siempre la primera dentro de una aplicación que utilice *JDBC*.

CONSULTAS

El esquema más sencillo para realizar una consulta a la base de datos con JDBC requiere la utilización de las siguientes clases e interfaces.

- La clase *DriverManager*
- La interfaz *Connection*
- La interfaz *Statement*
- La interfaz *ResultSet*

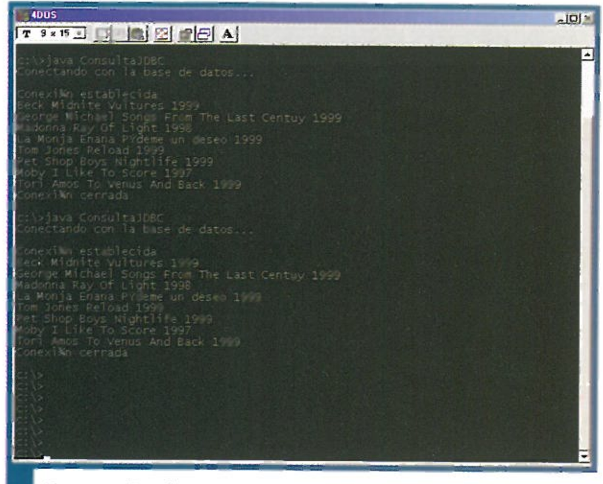


Figura 5.- Resultado de ejecutar la aplicación *ConsultaJDBC.java*

La clase *DriverManager* permite establecer la conexión con la con una base de datos. Las particularidades de la conexión quedan recogidas por un objeto de tipo *Connection*. La interfaz *Statement* -o *PrepareStatement*, o *CallableStatement*, como ya veremos más adelante- permite crear y ejecutar consultas *SQL* contra la base de datos y sobre la conexión proporcionada la interfaz *Connection*. Finalmente, la informacion que devuelve una consulta es accesible mediante un objeto que soporta la interfaz ResultSet.

LA CLASE DRIVERMANAGER

- *public static synchronized Connection getConnection(String url) throws SQLException*

Este es el primero de los métodos que podemos utilizar para llevar a cabo la conexión con la base de datos. Se pasa como único parámetro la dirección *URL JDBC* de la base de datos.

- *public static synchronized Connection getConnection(String url, String user, String password) throws SQLException*

En el caso en que sea necesario un usuario y una contraseña para acceder a la base de datos utilizaremos este método.

LA INTERFAZ CONNECTION

- *public abstract Statement createStatement() throws SQLException*

Las consultas *SQL* se ejecutan normalmente mediante la utilización de objetos del tipo *Statement*.

- *public abstract void close() throws SQLException*

Es conveniente cerrar las conexiones a bases de datos tan pronto como dejen de utilizarse, para liberar recursos rápidamente. Sin embargo, ha de tenerse en cuenta que establecer una conexión es una operación lenta, por lo que tampoco se debe estar abriendo y cerrando la conexión con frecuencia.

LA INTERFAZ STATEMENT

Permite ejecutar sentencias *SQL* sin parámetros.

- *public abstract ResultSet executeQuery(String sql) throws SQLException*

Ejecuta una sentencia SELECT y devuelve el resultado mediante la interfaz *ResultSet*.

- *public abstract int executeUpdate(String sql) throws SQLException*

Ejecuta una sentencia *UPDATE, DELETE, INSERT* o cualquier otra sentencia SQL que no devuelva un conjunto de resultados, y devuelve el número de registros afectados por la sentencia (o -1 si no los hubo).

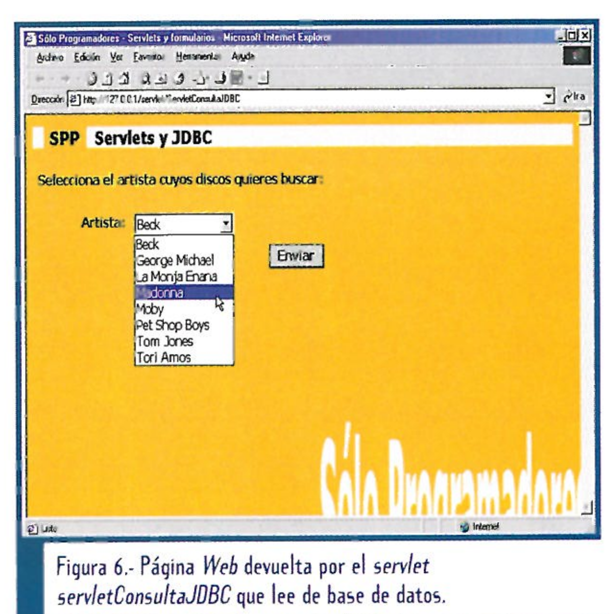


Figura 6.- Página *Web* devuelta por el *servlet* *servletConsultaJDBC* que lee de base de datos.

- *public abstract ResultSet getResultSet() throws SQLException*

Devuelve el *ResultSet* actual de la sentencia, si lo tiene.

- *public abstract void setMaxRows(int max) throws SQLException*

Establece el número máximo de registros que puede devolver *executeQuery*.

- *public abstract int getMaxRows() throws SQLException*

Devuelve el número máximo de registros que puede devolver *executeQuery*. El 0 representa un número ilimitado.

- *public abstract void setQueryTimeout(int seconds) throws SQLException*

Establece el tiempo en segundos que el controlador esperará hasta que el sistema gestor de bases de datos devuelva un resultado.

- *public abstract int getQueryTimeout() throws SQLException*

Devuelve el tiempo en segundos que el controlador esperará hasta que el sistema gestor de bases de datos devuelva un resultado. El 0 representa un tiempo ilimitado.

- *public abstract void close() throws SQLException*

Libera los recursos asociados a la consulta.

LA INTERFAZ RESULTSET

Un objeto de tipo *ResultSet* no es más que una tabla con los datos que devuelve la consulta *SQL* que se ha lanzado contra la base de datos. Esta tabla se recorre por medio de un puntero que inicialmente se encuentra apuntando justo a la posición inmediatamente anterior a la primera fila. El método *next* mueve el cursor a la siguiente posición, o lo que es lo mismo, a la siguiente fila de la tabla de resultados.

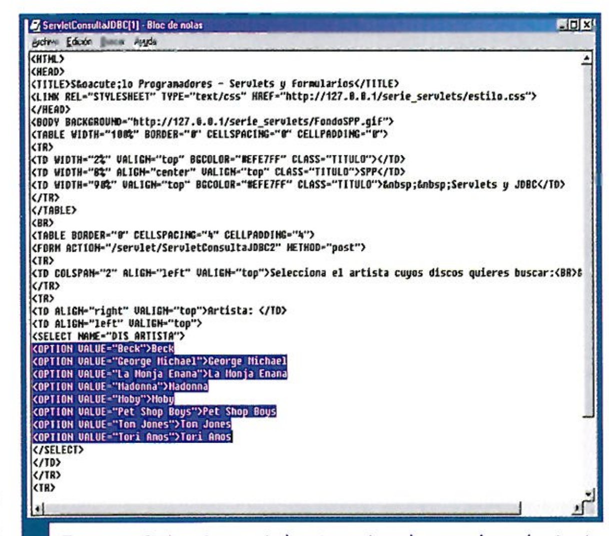
La interfaz *ResultSet* cuenta con un conjunto de métodos mediante los cuales se pueden obtener los valores almacenados en las columnas. Estos métodos se llaman *getXXX*, donde *XXX* representa el tipo de datos que se pretende leer. Por ejemplo, si la columna almacena una cadena de caracteres:

```
public String getString(int
    columnIndex)
    throws
    SQLException
public String getString(String
    columnName)
    throws
    SQLException
```

Como se puede observar los valores pueden ser accedidos utilizando un número que indica la posición de la columna dentro de la fila, o bien mediante el nombre de la misma. Por lo general, utilizar el índice siempre será más eficiente que hacerlo por el nombre.

Los controladores para protocolos de red independientes traducen las llamadas JDBC a un protocolo de red independiente de las bases de datos

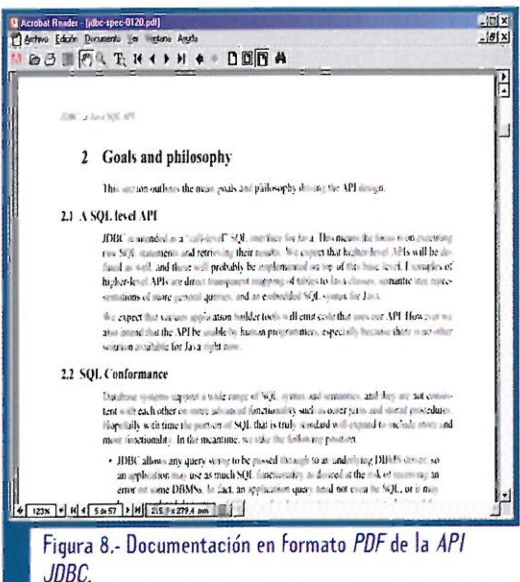
Los métodos *getXXX* no distinguen entre mayúsculas y minúsculas en lo que respecta a los nombres de las columnas que reciben como parámetro. Cuando hay varias columnas que tienen el mismo nombre se devuelve la primera que se encuentre. Esta forma de acceso está especialmente destinada para aquellos casos en los que las columnas son nombradas explícitamente en la consulta *SQL*. Si ese no es el caso, lo mejor es


ServletConsultaJDBC[1] - Bloc de notas

```
<HTML>
<HEAD>
<TITLE>S&oacute;lo Programadores - Servlets y Formularios</TITLE>
<LINK REL="STYLESHEET" TYPE="text/css" HREF="http://127.0.0.1/serie_servlets/estilo.css">
</HEAD>
<BODY BACKGROUND="http://127.0.0.1/serie_servlets/FondoSPP.gif">
<TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
<TR>
<TD WIDTH="2%" VALIGN="top" BGCOLOR="#EFE7FF" CLASS="TITULO"></TD>
<TD WIDTH="8%" ALIGN="center" VALIGN="top" CLASS="TITULO">SPP</TD>
<TD WIDTH="90%" VALIGN="top" BGCOLOR="#EFE7FF" CLASS="TITULO">  Servlets y JDBC</TD>
</TR>
</TABLE>
<BR>
<TABLE BORDER="0" CELLSPACING="4" CELLPADDING="4">
<FORM ACTION="/servlet/ServletConsultaJDBC2" METHOD="post">
<TR>
<TD COLSPAN="2" ALIGN="left" VALIGN="top">Selecciona el artista cuyos discos quieres buscar:<BR>&
</TR>
<TR>
<TD ALIGN="right" VALIGN="top">Artista: </TD>
<TD ALIGN="left" VALIGN="top">
<SELECT NAME="DIS_ARTISTA">
<OPTION VALUE="Beck">Beck
<OPTION VALUE="George Michael">George Michael
<OPTION VALUE="La Monja Enana">La Monja Enana
<OPTION VALUE="Madonna">Madonna
<OPTION VALUE="Moby">Moby
<OPTION VALUE="Pet Shop Boys">Pet Shop Boys
<OPTION VALUE="Tom Jones">Tom Jones
<OPTION VALUE="Tori Amos">Tori Amos
</SELECT>
</TD>
</TR>
<TR>
```

Figura 7.- Código fuente de la página devuelta por el *servlet* donde aparece seleccionado el código que se genera dinámicamente.




Acrobat Reader - [jdbc-spec-0120.pdf]

2 Goals and philosophy

2.1 A SQL level API

2.2 SQL Conformance

Figura 8.- Documentación en formato *PDF* de la *API JDBC*.

acceder a las columnas por medio de su índice.

Las columnas de una tabla de resultados deben ser leídas de izquierda a derecha en orden, y cada fila solamente debe ser leída una vez. Cuando se utiliza uno de los métodos *getXXX*, el controlador *JDBC* intenta siempre convertir el valor devuelto por la base de datos al correspondiente tipo *Java*.

Un objeto de tipo ResultSet no es más que una tabla con los datos que devuelve la consulta SQL que se ha lanzado contra la base de datos

Un objeto de tipo *ResultSet* se cierra automáticamente en el mismo momento que se cierra el objeto de tipo *Statement* que, al ejecutarse, lo generó.

El número, los tipos y las propiedades de las columnas de una tabla de resultados pueden ser consultados mediante la utilización de la interfaz *ResulSetMetaData*. El método *getMetaData* de un objeto de tipo *ResultSet* devuelve un objeto de tipo *ResulSetMetaData*.

Además de los métodos anteriores la interfaz proporciona otros muchos entre los que destacamos:

- *public abstract int findColumn(String columnName) throws SQLException*

  Devuelve el número de columna cuyo nombre se pasa como parámetro.

- *public abstract boolean next() throws SQLException*

  Utilizamos este método para recorrer el conjunto de resultados. Devuelve un valor que indica si existe otro registro delante del actual.

- *public abstract boolean wasNull() throws SQLException*

  Indica si el contenido de la última columna accedida es NULL.

- *public abstract void close() throws SQLException*

  Libera los recursos asociados al ResultSet.

EXCEPCIONES SQL

Finalmente vamos a echar un vistazo a las excepciones proporcionadas por el paquete *java.sql:*

- *DataTruncation*

  Este tipo de excepciones se producen cuando se trunca algún dato, ya sea al leerlo o al escribirlo. Por ejemplo al intentar almacenar un texto que excede la longitud de un campo.

- *SQLException*

  Este tipo de excepciones se producen cuando se produce un error al acceder a la base de datos.

Cada excepción *SQLException* proporciona la siguiente información:

- Una mensaje que describe el error y que es accesible mediante el método *getMessage.*
- Una cadena del tipo *SQLstate* que sigue la especificación *XOPEN SQLstate.*
- Un código de error que dependerá del fabricante

El gestor de controladores JDBC es la capa de software que se encarga de gestionar todos los controladores JDBC del sistema

Es posible que se encadenen varias excepciones, motivo por el que se proporciona el método get*NextException.*

ACCESO A UNA BBDD DESDE UN SERVLET

Antes de pasar a ver un ejemplo en el que se integra en un *servlet* el acceso a bases de datos, vamos a estudiar el código fuente de una pequeña aplicación *Java* que lee unos datos de una base de datos Access y los muestra por la pantalla.

Para poder ejecutar este ejemplo debemos configurar la base de

datos *Access* como un origen de datos *ODBC* dentro de *Windows*. Para ello haremos clic en *Inicio* > *Configuración* > Panel de control y seleccionaremos Fuentes de datos *ODBC* (32 bits). Entonces se abre el administrador de orígenes de datos *ODBC*. Seleccionamos la pestaña *DSN* Sistema y hacemos clic en el botón *Agregar*. En la primera ventana se solicita el tipo de controlador asociado a dicha base de datos. En el ejemplo que nos ocupa será *Microsoft AccessDriver (*.mdb)*. Por último será preciso dar un nombre al origen de datos y seleccionar el archivo con extensión *MDB* correspondiente a la base de datos.

La variable conexion es un objeto de tipo Connection que se obtiene a partir del método getConnection de la clase DriverManager

El archivo *ConsultaJDBC.java* contiene el código fuente de la aplicación que pasaremos a analizar a continuación. Se comienza creando una variable de tipo *String* con la dirección *URL JDBC* de la base de datos.

```
String urlBD = "jdbc:odbc:COLEC-
    CIONDISCOS";
```

El siguiente paso consiste en cargar la clase encargada de soportar el controlador *JDBC* necesario para acceder a la base de datos.

```
Class.forName("sun.jdbc.odbc.Jdb-
    cOdbcDriver");
```

La variable conexion es un objeto de tipo *Connection* que se obtiene a partir del método *getConnection* de la clase *DriverManager*. Este método recibe como parámetro la dirección *URL JDBC* de la base de datos contenida en la variable *urlBD*.

```
Connection conexion =
    DriverManager.get-
    Connection(urlBD);
```

Una vez que la conexión con la base de datos se ha producido, podemos pasar a crear sentencias *SQL* y ejecutarlas.

```
Statement select = cone-
    xion.createState-
    ment();
ResultSet resultadoSelect =
    select.executeQuery("SELECT *
    FROM DISCO");
```

El resultado de la ejecución de una de esas sentencias queda recogido en un objeto del tipo *ResultSet*. La información devuelta puede ser leída gracias a los métodos que proporciona esta interfaz.

```
boolean seguir = resultadoSe-
    lect.next();
while (seguir) {
  System.out.print(resultadoSe-
    lect.getString(2) + " ");
  ...
  seguir =
    resultadoSelect.next();
}
```

Por último, liberamos los recursos utilizados para conectarnos a la base de datos, crear una sentencia *SQL*, ejecutarla y leer la respuesta.

```
resultadoSelect.close();
select.close();
conexion.close();
```

Durante todas las fases anteriores pueden producirse errores. Éstos pueden ser capturados de forma que sea posible mostrar por la pantalla la información suficiente como para conocer los motivos del error.

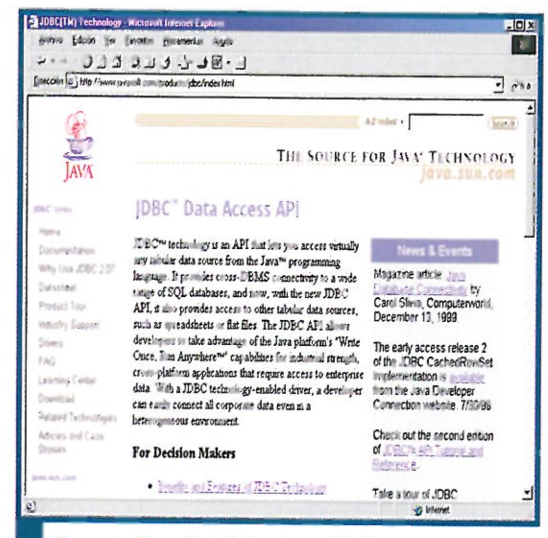
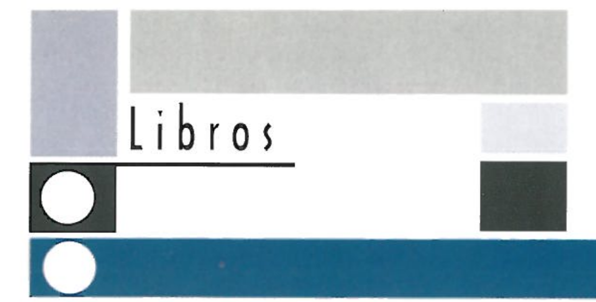


Figura 9.- Sitio *Web* dedicado a *JDBC* de *SUN* (www.javasoft.com/products/jdbc/index.htlm).

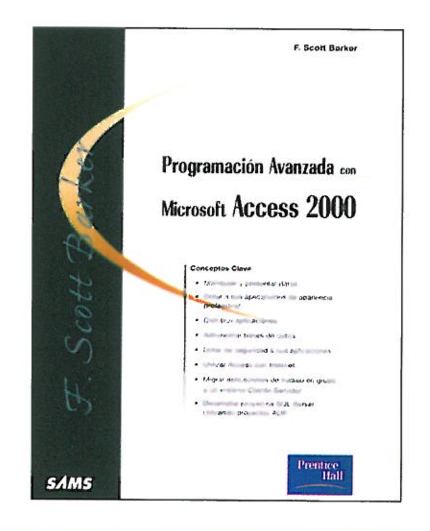
```
} catch(SQLException e) {
  System.out.println("Error:
    SQLException");
  while (e != null) {
    System.out.println("SQLState:
    " + e.getSQLState());
    System.out.println("Mensaje:"
    + e.getMessage());
    System.out.println("Vendor:"
    + e.getErrorCode());
    e = e.getNextException();
    System.out.println("");
  }
}
```

En principio, el esquema anterior es el mismo tanto si desarrollamos una aplicación como si es un *servlet*. En capítulos sucesivos de esta serie veremos que no es exactamente así pero por el momento podemos realizar esta consideración. El archivo *ServletConsultaJDBC.java* contiene el código fuente de un servlet que presenta una página *HTML* construida dinámicamente a partir de la información procedente de la base de datos.



PROGRAMACIÓN AVANZADA CON MICROSOFT ACCESS 2000

Conozca soluciones, proyectos y aplicaciones del mundo real para sus propios programas y así convertirse en un desarrollador de *Access* 2000 más eficiente y productivo.



Este libro le proporciona todo lo necesario para acceder al siguiente nivel de programación avanzada. En sus páginas descubrirá el control de aplicaciones *Office* mediante *DDE* y los mecanismos de automatización, utilizará la *API* para ampliar la potencia de *Access*, creará sus propios asistentes y complementos, aprenderá a gestionar entornos multiusuario.

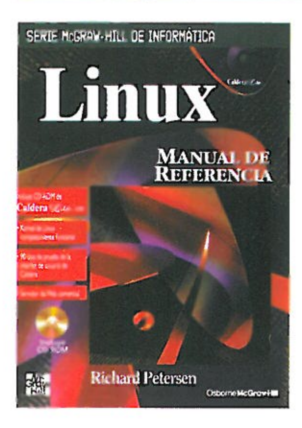
En sus páginas encontrará las ventajas e inconvenientes de los objetos de datos *ActiveX (ADO),* así como el uso de páginas de acceso a datos para publicar información en la *Web* o el modo de ampliar la potencia de su base de datos *Access* utilizando *SQL Server* como módulo de servicio.

Se trata de un manual orientado tanto para el desarrollador que trabaja en la empresa como para el consultor independiente, ofreciendo como objetivo común el hecho de diseñar un sistema robusto en el menor tiempo posible.

Editorial: Prentice Hall
Número de páginas: 1307
Nivel: Avanzado

Autor: F. Scott Barler
Idioma: Español
Precio: 7.995 Ptas. (I.V.A inc.)

LINUX MANUAL DE REFERENCIA



¿Está usted preparado para experimentar toda la potencia de *Linux*? Con esta completa guía de referencia no requiere ningún reconocimiento previo del popular sistema operativo. Tanto si es usuario novel como experimentado, este libro le introducirá rápidamente en el apasionante universo *Linux*.

Con este manual se comienza aprendiendo los fundamentos de la propia instalación del sistema para, paso a paso, llegar a conocer las características más avanzadas, incluyendo la resolución de problemas típicos y conceptos sobre programación y comunicaciones.

Escrito de forma amena y sencilla, podrá obtener todas las ventajas del trabajo en red a través de *Linux*, llegar a ser un experto en conexiones *Internet* con este sistema tales como *FTP, Telnet, Gopher, mail, news* y *Web*, así como crear servidores *Web* utilizando este sistema operativo.

El libro incluye un *CD-Rom* de *Caldera Lite* con el *kernel* de *Linux* completamente funcional y 90 días de prueba con la interfaz de usuario de *Caldera* que le ayudará a sumergirse en este sistema sin ningún problema, demostrándole que cualquiera puede convertirse en un experto siguiendo tan sólo las explicaciones claras y detalladas que ofrece este libro.

Editorial: McGraw Hill
Nº de páginas: 5894
Nivel: Intermedio

Autor: Richard Petersen
Idioma: Español
Precio: 8.000 Ptas. (I.V.A inc.)

DHTML HTML DINÁMICO



Este libro le proporcionará la información más actualizada respecto al grupo de tecnologías que abarca el revolucionario descubrimiento conocido como *HTML* Dinámico.

Conseguirá aprender a dominar los aspectos más complejos de las Hojas de Estilo en Cascada *(Cascading Style Sheets)* y del posicionamiento de *HEC (CSS Positioning),* la maquetación *3D* y la asignación de formato y contenidos *(Content Formatting),* unas técnicas comprensibles de *JavaScript*, para crear unos contenidos *Web* desarrollados profesionalmente, mediante el diseño de *HTML* Dinámico.

Hallará un completo tratamiento de las características de *HTML* Dinámico tal como están implementadas tanto en *Microsoft Internet Explorer* 4 como en *Netscape Communicator.*

Se pondrá al día en lo tocante a contenido interactivo, integración de bases de datos sin discontinuidad aparente, y capacidades de maquetación dinámica que se pueden personalizar en el momento de la carga o durante la ejecución para maximizar la eficiencia del servidor o proporcionar al usuario una experiencia personalizada de la *Web*.

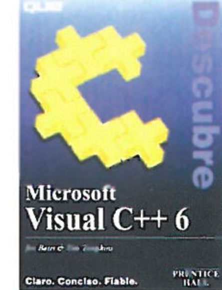
Con esta Edición Especial, será capaz de construir contenidos automodificables para acomodarse al navegador del usuario y a los parámetros del sistema, incrementar la interactividad con manejo de sucesos y una gestión de animaciones realizadas localmente por parte del cliente, así como producir efectos visualmente sorprendentes y fácilmente programables mediante filtros y transiciones.

Editorial: Prentice Hall
Número de páginas: 732
Nivel: Intermedio
Autor: Gulbransen y Rawlings
Idioma: Español
Precio: 6.180 Ptas. (I.V.A. inc.)

DESCUBRE MICROSOFT VISUAL C++ 6

Este es el momento de aprender a diseñar, programar y ejecutar aplicaciones con *Visual C++* 6, y este manual le resultará indispensable a la hora de ponerse manos a la obra en esta labor.

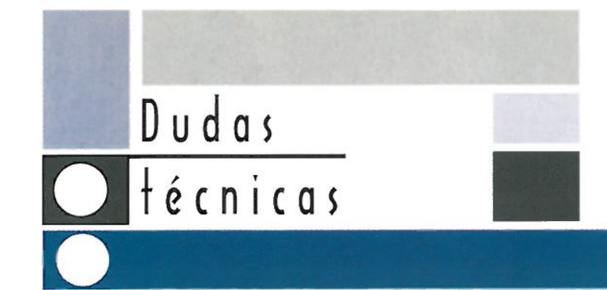
Gracias a este libro, comprenderá la programación *OLE* y *COM*, creará y trabajará con controles *ActiveX*, dominará otro tipo de controles tales como *List Control*, *Date Time Control, Progress Control* y muchos más. Será capaz de utilizar *Debbuger* y *Profiler* y de explorar documentación avanzada y técnica de visualización.



Gracias a la claridad de las explicaciones el lector no encontrará ningún problema a la hora de asimilar conceptos y términos a lo largo de estas páginas, consiguiendo realizar las tareas encomendadas de forma sencilla al seguir paso a paso las instrucciones que se van dando.

Con un rápido vistazo será capaz de encontrar información adicional como trucos y atajos que puedan mejorar su rendimiento en el trabajo con estas aplicaciones. Este volumen supone una referencia imprescindible a la hora de sacarle el máximo partido a *Microsoft Visual C++* 6.

Editorial: Prentice Hall
Nº de páginas: 706
Nivel: Intermedio
Autor: Bases y Tornpkins
Idioma: Español
Precio: 5.995 Ptas. (I.V.A. inc.)



Dudas técnicas

En esta sección, como cada mes, SÓLO PROGRAMADORES os brinda la oportunidad de encontrar respuesta a las dudas que podáis tener, en cualquier tema relacionado con la programación y bajo cualquier entorno de desarrollo. Ya sabéis que nuestra dirección es solop@virtualsw.es

PREGUNTA

Hola amigos:

Me dirijo a vosotros como último recurso para tratar de localizar el Personal *Web Server* para *Windows* 95 que se refleja en el artículo sobre *ASP* de la revista *Sólo Programadores.*

He buscado en la *Web* de *Microsoft* y encontré el Personal *Web Server* 4.0 como parte del *Option Pack* para *Windows NT* 4.0. Sin embargo no puedo terminar la descarga y tampoco sé si es ése el que necesito para poder seguir el curso de *ASP* que se está publicando. Le agradecería que me indicara otro lugar desde donde poder descargar el *PWS*.

Reciban un cordial saludo.

Angel García Camacho.

RESPUESTA

Estimado lector:

Evidentemente para seguir el curso sobre *ASP* necesitas tener un servidor y que además sea capaz de ejecutar este tipo de páginas. La solución más sencilla, cuando la finalidad es didáctica y hablamos de un ordenador personal con *Windows*, es sin duda alguna *PWS* (*Personal Web Server*). En el propio *CD-Rom* de *Windows* 98 se encuentra el instalador de este servidor, pero con *Windows* 95 no queda más remedio que descargar el *Option Pack* del sitio *Web* de *Microsoft* e instalarlo. Ahora bien, es probable que todo funcione sin problemas si instalas la versión del *CD-Rom* de *Windows* 98 sobre *Windows* 95. También puedes escribir al soporte de *Microsoft* y preguntarles acerca de la posibilidad de que te envíen el *Option Pack* en un *CD-Rom*.

PREGUNTA

Estimados amigos de *Sólo Programadores*:

En primer lugar, daros mi más sincera enhorabuena por la revista. Personalmente opino que conformáis la mejor revista de programación. Por cierto, la *suite* de *Internet-Visual Basic* (de Jordi Agost) me parece fabulosa.

Tengo diversas dudas en diferentes campos, así que espero que me podáis ayudar con todas ellas.

VISUAL BASIC

- Una vez realizado un programa en *Visual Basic,* ¿qué es lo que tendría que hacer para que cuando el usuario ejecute el programa de instalación, aparezca como en cualquier programa un número de serie? Y por tanto, si no lo supiera que se interrumpe la instalación de dicho programa mientras que si lo supiera, que continuase con el proceso.

- ¿Qué es lo que tendría que hacer para conseguir que tras pulsar en un botón *(command*

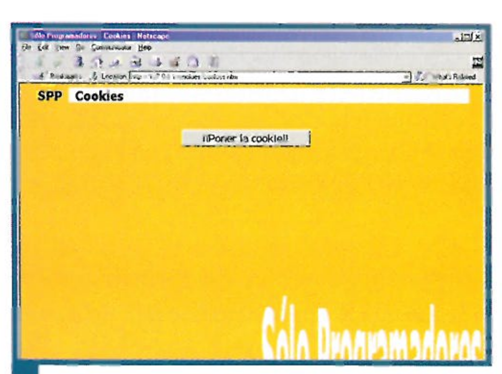


Figura 1.- Cuando la *cookie* no existe se muestra un botón mediante el cual puede ponerse

button) me salga la suma de todos los registros (la suma de números, ya que son numéricos) contenidos en el campo por ejemplo, *PVP*, de una base de datos?

HTML

- Tengo mucho interés en saber cómo se puede crear un contador (propio) sencillo para una página *Web* (para controlar cuántos han accedido a mi sitio *Web*) pero con la característica de que empiece, por ejemplo a partir de 1000 visitas.

JAVASCRIPT

- Una vez pulsado en un botón de opción, me gustaría saber cómo podría salir un mensaje (utilizando la función por ej.: *alert*) y se oculte o se desactive dicho botón de opción para que no pueda volver a pulsarlo de nuevo. Es para un cuestionario que estoy haciendo y una vez elegido una respuesta que no pueda volver a pulsarla de nuevo.

- Me gustaría saber cómo puedo limitar en una caja de texto (de una pág. *HTML*) a que el usuario sólo pueda introducir números. Y si el usuario ha introducido letras que automáticamente salte un mensaje de información de formato incorrecto.

Figura 2.- Cuando la *cookie* existe entonces se muestra un mensaje. También se visualiza un botón mediante el cuál podemos quitarla.

- ¿Cómo podría comprobar, tras pulsar en un botón (*command button)* si se ha escrito o no en una caja de texto por ejemplo, la palabra *“ordenador”* y el usuario ha escrito en verdad la palabra *“avordenadorcd”*;, así como cualquier combinación posible?

Muchas gracias. Mientras tanto espero ansiosa vuestras respuestas. Un saludo para todos los que hacéis la magnífica revista *Sólo Programadores*

Ángeles Benítez. Sevilla.

RESPUESTA

Estimada lectora:

Vamos a intentar contestar una a una todas las preguntas de tu extenso mensaje. Pero antes de empezar bien vale un "gracias" por la atención que nos dispensas.

VISUAL BASIC

- Programas con número de serie.

Existen diversos métodos para proteger los programas con un número de serie. Algunos son muy sencillos y ofrecen muy poca protección y otros son más robustos. Por ejemplo: puedes utilizar el registro de *Windows*. Localizas una subclave cuyo nombre no ofrezca demasiadas pistas y guardas allí la información relativa al estado actual del programa: si está registrado o no, y si lo está cuál es el número de serie.

Las rutinas encargadas de aceptar y validar el código pueden estar contenidas en el propio código del programa que distribuyes o en una librería a parte. Puedes tener una lista de números de serie válidos o bien una rutina que sea capaz de codificar/decodificar los números de serie dinámicamente.

La primera de la soluciones presenta dos inconvenientes: los números de serie, aunque los almacenes en un ejecutable o en una librería, deberían estar codificados ya que podría ser relativamente fácil extraerlos; además, al tener una cantidad limitada de números de serie podrías tener problemas de distribución. La segunda de las soluciones es algo mejor aunque en ese caso tienes que desarrollar unas rutinas de codificación/decodificación lo suficientemente buenas como para que no pueda deducirse su comportamiento fácilmente a partir de los códigos que se aceptan.

- Acceso a una base de datos.

Todo depende de la base de datos a la que quieras acceder, pero si se trata por ejemplo de un archivo *MDB* de *Access*, o incluso de un archivo de *Excel*, en realidad lo único que tienes que hacer es abrir la base de datos, leer los registros mediante un bucle y realizar la suma. No reviste la menor dificultad. Consulta la ayuda de *Visual Basic* y a buen seguro encontrarás un ejemplo sencillo de cómo hacerlo. En cualquier caso te recomendamos que visites *www.vb-zone.com* y *msdn.microsoft.com/vbasic/technical/tips.asp*. Estos son sitios *Web* donde a buen seguro encontrarás varios ejemplos a partir de los cuales podrás desarrollar el tuyo propio.

HTML

- Contador para las páginas *Web*.

Los contadores de visitantes para páginas *Web* pueden ser desarrollados en multitud de tecnologías. Pero tienes que ser consciente de que sea cual sea la elegida, siempre hablamos de una tecnología de servidor. Por ejemplo: programas *CGI* en *Perl* o en *C*, páginas *ASP*, *ColdFusion*, *Java Server Pages*, *Java Servlets*, etc. Por ello, todo depende directa-

mente del servidor donde estén alojadas tus páginas *Web*. Lo primero que debes hacer es preguntar al *webmaster*. Con frecuencia suelen ser muy restrictivos con respecto a los programas que un usuario puede depositar en el servidor, pero a veces también proporcionan algunos programas *CGI*, que han sido desarrollados por ellos mismos, y que sirven para cosas tales como contadores de visitas.

JAVASCRIPT

● Botón con memoria.

Tal y como nos planteas tu problema la única solución posible que tienes consiste en la utilización de *cookies* con *Javascript*. Para ello puedes utilizar tres funciones como las que siguen:

```
function ponerCookie(nombre,
    valor, caducidad) {
 document.cookie = escape(nombre) +
                   "=" +
                   escape(valor) +
                   "; expires= " +
                   caducidad.
                    toGMTString();
}
function eliminarCookie(nombre) {
  var fecha = new Date();
 document.cookie = escape(nombre) +
                   "=" +
                   escape("-") +
                   "; expires= " +
                   fecha.
                    toGMTString();
}
function leerCookie(nombre) {
  var cadena_a_buscar = nombre +
                        "=";
  var longitud = cadena_a_
                 buscar.length;
  var posicion_desde =
    document.cookie.indexOf
      (cadena_a_buscar);
  var posicion_hasta;

  if(posicion_desde != -1) {
    posicion_desde += longitud;
    posicion_hasta =
     document.cookie.indexOf(";",
      posicion_desde);
    if(posicion_hasta == -1) {
      posicion_hasta =
        document.cookie.length;
    }
    return unescape(document.
     cookie.substring(posicion_
      desde, posicion_hasta));
  }
  return null;
}
```

Una vez que tienes estás funciones, puedes utilizarlas de la siguiente forma dentro de la página.

Si la *cookie* no existe entonces se mostrará el botón para ponerla. En otro caso se muestra un mensaje diciendo que la *cookie* existe y además se proporciona un bo-tón para quitarla. Puedes experimentar con diferentes versiones de estas funciones hasta que des con el comportamiento que buscas.

● Sólo números.

Lo que tienes que hacer es una sencilla función de *Javascript* que se active cuando se apriete el botón de *submit* del formulario. Por ejemplo:

```
<FORM ··· onsubmit="return
      Validar(this)">
```

La función validar podría llamar a su vez a otra función que se encargara de verificar si realmente es número lo que ha introducido el usuario.

```
function EsNumero(cadena) {
  var numeros = "0123456789";
  var i, longitud;

  longitud = cadena.length;
  for(i=0;i<longitud;i++) {
    if (numeros.indexOf(cadena.
     charAt(i)) == -1) {
      return false;
    }
  }
  return true;
}
function Validar(f) {
  if (!EsNumero(f.CAMPO.value)) {
    alert("No es un número");
    f.CAMPO.focus();
    f.CAMPO.select();
    return false;
  }
  return true;
}
```

● Validación de formularios.

El tercero de los casos del que nos hablas es similar al anterior. Lo único que tendrías que hacer es seguir añadiendo comprobaciones a la función *Validar* para verificar que todo es correcto antes de que el formulario se mande.

```
<SCRIPT LANGUAGE="JavaScript">
<!--
function PonerCookie() {
 var fecha = new Date(2000, 10, 1);
 ponerCookie("micookie",
  fecha.toGMTString(), fecha);
 window.location.reload();
}
function QuitarCookie() {
 eliminarCookie("micookie");
 window.location.reload();
 }
 if (leerCookie("micookie") == null) {
  document.write('<INPUT TYPE=
   "button" NAME="PONER" VALUE=
     "¡¡Poner la cookie!!" onclick=
      "PonerCookie()">');
 } else {
  document.write('<SPAN CLASS=
   "COOKIEPUESTA">');
  document.write('&iexcl;&iexcl;
    La cookie ya est&aacute; puesta!!');
  document.write('<BR> <BR>
    </SPAN>');
  document.write('<INPUT TYPE=
   "button" NAME="QUITAR" VALUE=
     "¡¡Quitar la cookie!!" onclick=
      "QuitarCookie()">');
 }
//-->
</SCRIPT>
```